

UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET



Miloš Milaković

SEKVENCIARANJE ANTIBIOTIKA -
ELEKTRONSKA LEKCIJA

master rad

Beograd, 2025.

Mentor:

dr Jovana KOVAČEVIĆ, vandredni profesor
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Mirjana MALJKOVIĆ RUŽIČIĆ, docent
Univerzitet u Beogradu, Matematički fakultet

Nevena ĆIRIĆ, asistent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Najvoljenijima

Naslov master rada: Sekvenciranje antibiotika - Elektronska lekcija

Rezime: Ovaj rad predstavlja elektronsku lekciju posvećenu sekvenciranju antibiotika, sa fokusom na interaktivno upoznavanje sa različitim algoritamskim pristupima. Lekcija obuhvata teorijsko objašnjenje i vizuelizaciju algoritama kao što su gruba sila, *Branch and Bound*, *Leaderboard* algoritam, spektralna konvolucija i *DeepNovo* sekvenciranje. Korisnicima je omogućeno da prate izvršavanje algoritama korak po korak, sa opcijama pauziranja i ponavljanja, čime se olakšava razumevanje kompleksnih procesa sekvenciranja. Ova interaktivna platforma može služiti kao edukativni alat za studente i predavače u oblasti bioinformatike.

Ključne reči: sekvenciranje, algoritmi, računarstvo, aminokiseline, maseni spektrometar, gruba sila, *Branch and Bound*, *Leaderboard*, *DeepNovo*, spektralna konvolucija

Sadržaj

1	Uvod	1
1.1	Cilj rada	2
2	Teorijske osnove	4
2.1	Centralna dogma molekularne biologije	4
2.2	Odstupanje od centralne dogme	6
2.3	Maseni spektrometar	7
2.4	Teorijski spektar peptida	9
3	Algoritmi za sekvenciranje	11
3.1	Gruba sila (<i>Brute Force</i>)	11
3.2	Branch and Bound	13
3.3	Uporedna analiza prethodnih algoritama	16
3.3.1	Algoritam grube sile	16
3.3.2	Algoritam Branch and Bound	17
3.3.3	Zaključak	17
3.4	Leaderboard algoritam	18
3.4.1	Prednosti algoritma	18
3.4.2	Primene	18
3.5	Spektralna konvolucija	20
3.6	DeepNovo	23
3.6.1	Računska složenost	23
3.6.2	Tehnike zasnovane na De Novo sekvenciranju	23
3.6.3	Opšti pregled	24
3.6.4	Arhitektura neuronske mreže	25
3.6.5	Rezultati i evaluacija	27

4	Elektronska platforma	29
4.1	Pokretanje aplikacije	29
4.1.1	Docker Compose konfiguracija	29
4.1.2	Direktno pokretanje komponenti	31
4.1.3	Google Cloud Run implementacija	33
4.2	Funkcionalnosti platforme	36
4.2.1	Početna stranica	36
4.2.2	Uvodna stranica	37
4.2.3	Algoritam grube sile	39
4.2.4	Branch and Bound	42
4.2.5	Leaderboard	43
4.2.6	Spektralna konvolucija	46
4.2.7	DeepNovo	48
4.2.8	Poređenje algoritama	49
4.2.9	Nepostojeća stranica	50
5	Zaključak	52
	Bibliografija	53

Glava 1

Uvod

Sekvenciranje predstavlja proces određivanja preciznog redosleda gradivnih jedinica bioloških molekula. Najpoznatiji oblik je sekvenciranje DNK, kojim se utvrđuje niz nukleotida u genomu organizma. Međutim, osim genoma, moguće je sekvencirati i proteine, pri čemu se određuje redosled aminokiselina koji čini njihovu primarnu strukturu. Takav postupak naziva se sekvenciranje proteina.

Za razliku od tehnike sekvenciranja genoma, koja koristi univerzalni genetski kod kao osnovu, određivanje strukture proteina zahteva eksperimentalne metode, najčešće masenu spektrometriju. U prirodi postoji 20 standardnih aminokiselina (slika 1.1) koje se kombinuju u različitim sekvencama kako bi formirale veliki broj funkcionalnih proteina. U kontekstu antibiotika, specifičan redosled aminokiselina je od presudnog značaja jer direktno utiče na:

- Trodimenzionalnu strukturu molekula
- Biološku aktivnost i mehanizam dejstva

Mnogi savremeni antibiotici su peptidnog porekla, što znači da su zapravo kratki lanci aminokiselina, tj. mali proteini. Antibiotici predstavljaju hemijska jedinjenja koja uništavaju mikroorganizme ili inhibiraju njihov rast, čime imaju ključnu ulogu u borbi protiv infekcija. Zbog njihove biološke važnosti i sve izraženijeg problema antimikrobne rezistencije, sekvenciranje antibiotika predstavlja značajnu oblast istraživanja u savremenoj bioinformatičkoj i farmaceutskoj industriji. Ovaj rad će se fokusirati upravo na proces sekvenciranja peptidnih antibiotika.

Aminokiselina	Skraćenica	Masa (Da)
Glycine	G	57
Alanine	A	71
Serine	S	87
Proline	P	97
Valine	V	99
Threonine	T	101
Cysteine	C	103
Isoleucine	I	113
Leucine	L	113
Asparagine	N	114
Aspartic Acid	D	115
Lysine	K	128
Glutamine	Q	128
Glutamic Acid	E	129
Methionine	M	131
Histidine	H	137
Phenylalanine	F	147
Arginine	R	156
Tyrosine	Y	163
Tryptophan	W	186

Slika 1.1: Tabela masa aminokiselina izraženih u daltonima (Da)

1.1 Cilj rada

U ovom radu biće predstavljena interaktivna elektronska platforma namenjena prikazu algoritamskih pristupa za sekvenciranje antibiotika, sa fokusom na njihovu vizuelizaciju u realnom vremenu. Platforma korisnicima omogućava da istraže principe i rad savremenih algoritama kroz interaktivne simulacije, čime se olakšava

razumevanje kompleksnih bioinformatičkih koncepata.

Glavni ciljevi ovog rada su:

- Pružiti pregled modernih algoritama za sekvenciranje antibiotika
- Razviti interaktivnu edukativnu platformu za vizuelizaciju sekvenciranja

Rad je posebno koristan studentima bioinformatike i molekularne biologije pružajući im alat za bolje razumevanje osnovnih principa masene spektrometrije i algoritama za rekonstrukciju peptidnih sekvenci.

Glava 2

Teorijske osnove

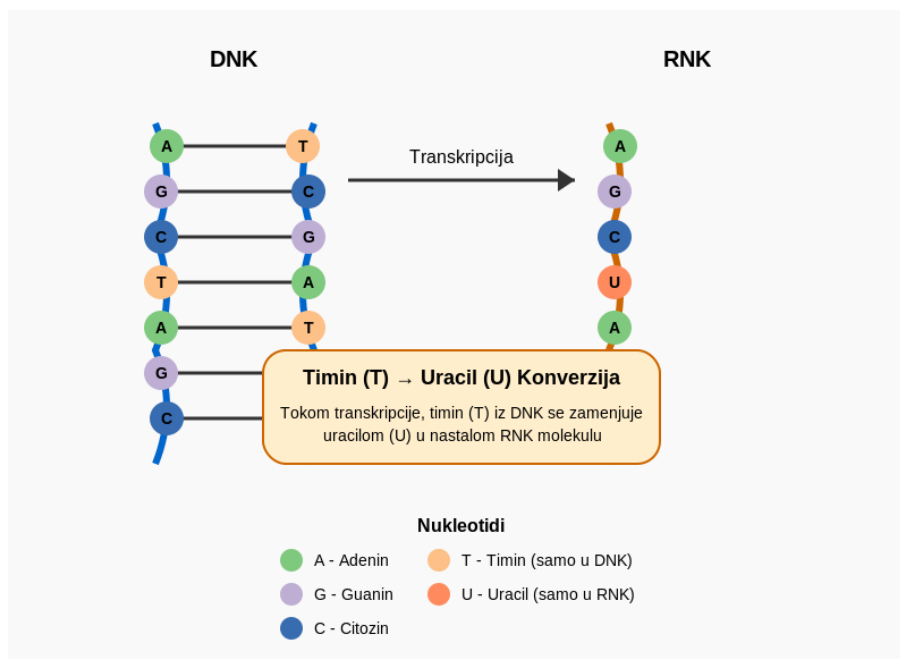
2.1 Centralna dogma molekularne biologije

Proces sekvenciranja antibiotika je fundamentalan u razumevanju kako su ovi molekuli proizvedeni od strane bakterija i kako se oni mogu sintetizovati ili modifikovani za primene u medicini. Antibiotici su često peptidi ali mnogo antibiotika, uglavnom neribozomalni peptidi (*non-ribosomal peptides* - *NRPs*), ne prati standardna pravila za sintezu proteina čime se otežava njihovo sekvenciranje [12, 14].

DNK sadrži recept za kreiranje proteina. Odnosno, sastoji se od gena koji mogu biti uključeni i tada će se na osnovu njih kreirati proteini ili isključeni kada se oni neće koristiti za kreiranje proteina. Ovaj proces uključivanja i isključivanja gena naziva se genska ekspresija i zavisi od toga da li je potrebno da se određeni protein kreira ili ne (na primer, fotosinteza kod biljaka koja se obavlja samo preko dana).

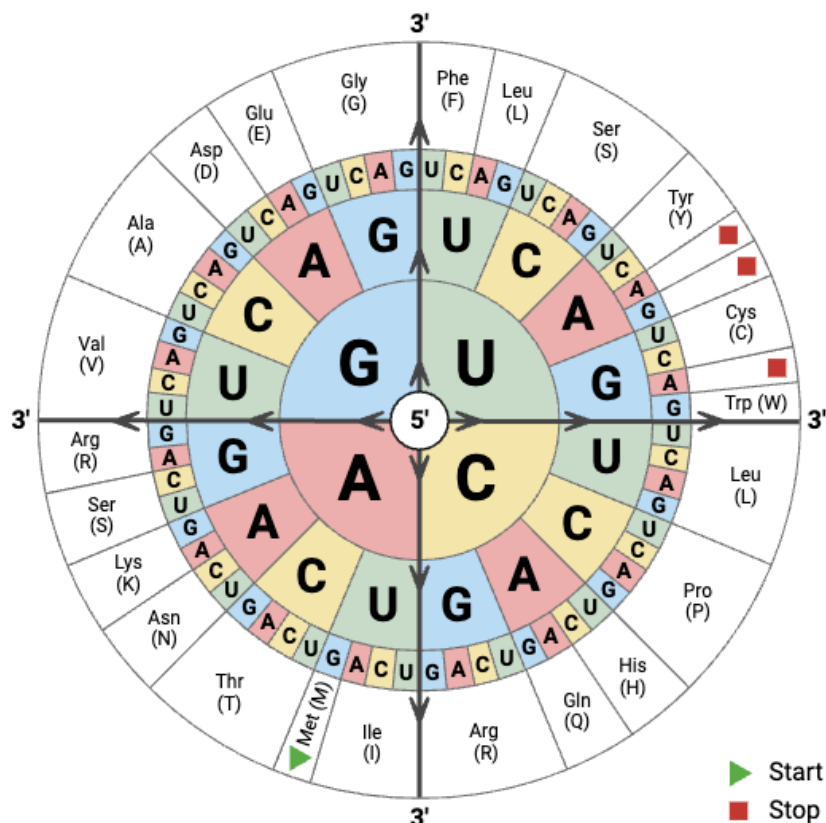
Tradicionalno, proteini prate Centralnu Dogmu Molekularne biologije, koja kaže da se DNK prvo prepisuje u RNK (slika 2.1), a zatim se RNK prevodi u protein. Proces prevođenja DNK u RNK naziva se transkripcija i on podrazumeva da enzim RNK polimeraza očitava jedan lanac DNK i na osnovu njega sintetiše jedan, komplementaran RNK lanac, pri čemu se nukleotid timin (T) zamenjuje uracilom (U). Na slici 2.1 se može se primetiti da se DNK sastoji od 2 lanca koja su komplementarna. Enzim RNK polimeraza se kači na početak gena i kreće kroz gene gde razdvaja lanac i stvara prostor za prepisivanje DNK u RNK čime se dobija RNK.

Prilikom prevođenja RNK u protein potrebno je na osnovu nukleotida odrediti koja je aminokiselina u pitanju. Za ovaj proces je zadužena organela ribozom, koja omogućava pravilno očitavanje informacija sa RNK. Budući da je potrebno uniformno tumačiti nukleotidnu sekvencu, koristi se niz od tri uzastopna nukleotida, poznat



Slika 2.1: Transkripcija DNK u RNK

kao kodon. Upotrebom kodona dužine tri nukleotida dobija se ukupno 64 različite sekvence, koje se mapiraju na 20 različitih aminokiselina. Da je korišćen niz od samo dva nukleotida, bilo bi moguće formirati svega 16 kombinacija, što ne bi bilo dovoljno da se pokriju sve aminokiseline neophodne za sintezu proteina. Na slici 2.2 može se videti kako se kodoni prevode u odgovarajuće aminokiseline. Postoje start i stop kodoni koji određuju početak odnosno kraj sekvence koja se prevodi u protein.



Slika 2.2: RNK kodonski točak prikazuje kako se sekvence od tri nukleotida (kodoni) prevode u aminokiseline. Svaki kodon se čita od centra ka spolja, a zeleni trougao označava start kodon (AUG) koji kodira metionin, dok crveni kvadrati označavaju stop kodone (UAA, UAG, UGA) koji određuju kraj sekvence koja se prevodi u protein. Preuzeto sa [15].

2.2 Odstupanje od centralne dogme

Tiroidin B1 je cikličan peptid dužine 10 (slika 2.3), što znači da su prva i poslednja aminokiselina povezane i da samim tim postoji 10 njegovih različitih linearnih reprezentacija, tabela 2.1. Prateći centralnu dogmu i zaključka da se 1 kodon prevodi u 1 aminokiselinu, naučnici su očekivali da pronađu 10 kodona odnosno 30 nukleotida u genomu bakterije *Bacillus brevis* od koje nastaje ovaj antibiotik. Da bi se ovaj postupak obavio, potrebno je proveriti više hiljada 30-grama koji mogu

da počnu bilo gde u genomu, što može delovati kao dugotrajan zadatak, ali uz moderne računare, ovaj proces se obavlja efikasno. Analiziranjem genoma utvrđeno je da ne postoji 30-gram koji se kodira u neki od 10 različitih reprenzacija traženog antibiotika.

#	Linearna sekvenca
1	Lys – Leu – Phe – Pro – Trp – Phe – Asn – Gln – Tyr – Val
2	Leu – Phe – Pro – Trp – Phe – Asn – Gln – Tyr – Val – Lys
3	Phe – Pro – Trp – Phe – Asn – Gln – Tyr – Val – Lys – Leu
4	Pro – Trp – Phe – Asn – Gln – Tyr – Val – Lys – Leu – Phe
5	Trp – Phe – Asn – Gln – Tyr – Val – Lys – Leu – Phe – Pro
6	Phe – Asn – Gln – Tyr – Val – Lys – Leu – Phe – Pro – Trp
7	Asn – Gln – Tyr – Val – Lys – Leu – Phe – Pro – Trp – Phe
8	Gln – Tyr – Val – Lys – Leu – Phe – Pro – Trp – Phe – Asn
9	Tyr – Val – Lys – Leu – Phe – Pro – Trp – Phe – Asn – Gln
10	Val – Lys – Leu – Phe – Pro – Trp – Phe – Asn – Gln – Tyr

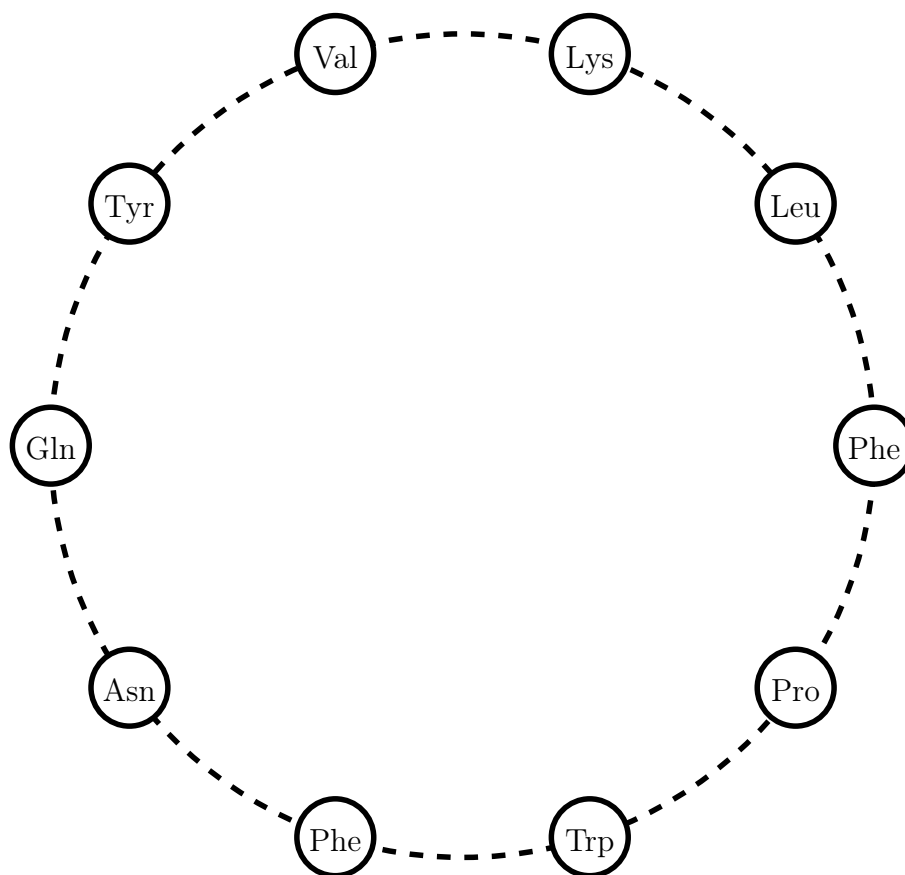
Tabela 2.1: Deset različitih linearnih reprezentacija tirocidina B1

Dokazano je da Tirocidin B1 ne prati centralnu dogmu molekularne biologije i da postoje posebni enzimi koji su zaduženi za njihovo sintentisanje. Oni se zovu *NRP* sintetaze. Ovi enzimi sadrže komplikovane module, koji govore koje aminokiseline učestvuju u sastavu proteina. U slučaju Tirocidina B1, enzim sadrži 10 modula i svaki od modula kodira 1 aminokiselinu čime je određena struktura antibiotika.

Samim tim, pošto struktura proteina nije određena na osnovu genoma bakterije, metode za sekvencioniranje DNK ovde nisu od pomoći i potrebno je sekvencirati direktno sam peptid.

2.3 Maseni spektrometar

Maseni spektrometar je moćan alat pomoću koga mogu da se odrede mase molekula, uključujući mase peptida i proteina. Omogućava naučnicima da odrede nepoznate komponente, saznaju strukturu molekula i analiziraju kompleksne uzorke. Maseni spektrometar radi tako što mu se da više uzoraka istog peptida a on napravi sve moguće podpeptide datog peptida i odredi njihove mase. U realnosti uzorak se pretvara u naelektrisane jone da bi na njih mogli da utiču električno i magnetno polje. Potom se joni dele na osnovu odnosa njihove mase i naelektrisanja i kao takvi se mere njihove vrednosti [11].



Slika 2.3: Struktura tirocidina B1, cikličnog peptida sastavljenog od 10 aminokiselina.

Masa se meri u daltonima (Da), pri čemu je 1 Da približno jednak masi protona/neutrona. Samim tim masa molekula je jednaka sumi masa protona/neutrona koji čine taj molekul. Mase aminokiselina su poznate i prikazane su na Slici 1.1. Može se primetiti da neke aminokiseline imaju istu masu, tako da 20 različitih aminokiselina ima 18 različitih masa.

Samim tim na osnovu poznatih masa aminokiselina možemo da odredimo da je masa tirocidina:

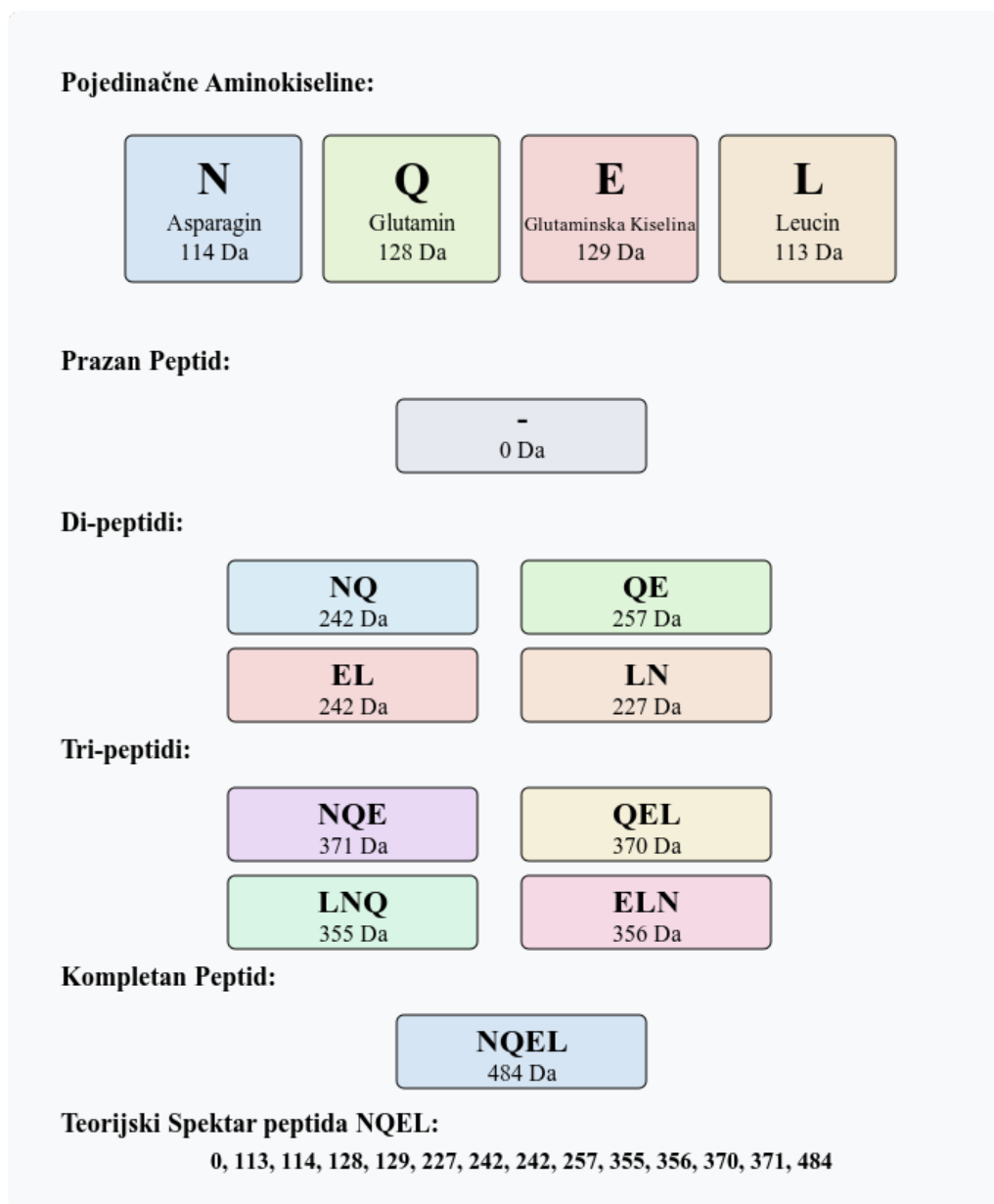
$$\begin{array}{cccccccccc} \text{V} & \text{K} & \text{L} & \text{F} & \text{P} & \text{W} & \text{F} & \text{N} & \text{Q} & \text{Y} \\ 99 & + & 128 & + & 113 & + & 147 & + & 97 & + & 186 & + & 147 & + & 114 & + & 128 & + & 163 & = & 1322 \end{array}$$

2.4 Teorijski spektar peptida

Teorijski spektar peptida predstavlja mase svih mogućih podpeptida, uključujući 0 i masu celog peptida. Na osnovu peptida možemo lako da odredimo teorijski spektar ali na osnovu spektra ne možemo lako da odredimo koji je peptid u pitanju.

Problem sekvenciranja ciklopeptida samim tim se svodi na problem kako rekonstruisati ciklični peptid na osnovu njegovog teorijskog spektra. U nastavku će biti prikazani nekoliko različitih algoritama.

Kao ulaz u svaki od ovih algoritama očekuje se eksperimentalni spektar, odnosno spektar koji je dobijen uz pomoć masenog spektrometra za neki peptid. Na Slici 2.4 su prikazane mase svih podpeptida peptida **NQEL** koje se dobijaju uz pomoć masenog spektrometra, kao i masa praznog peptida i celog peptida, takođe je prikazan i teorijski spektar.



Slika 2.4: Teorijski spektar peptida **NQEL** koji prikazuje sve moguće podpeptide, njihove mase i njegov teorijski spektar

Glava 3

Algoritmi za sekvenciranje

U ovom odeljku biće prikazani algoritmi za određivanje cikličnog peptida na osnovu poznatog eksperimentalnog spektra. Neki od algoritama koji će biti objašnjeni su:

- **Algoritam grube sile** (*Brute force*): Direktan pristup gde se isprobavaju sve moguće kombinacije da bi se našlo optimalno rešenje.
- **Branch and Bound**: Optimizovan algoritam koji će odbacivati kandidate čim prestanu da budu potencijalno rešenje.
- **Leaderboard algoritam**: Algoritam koji održava listu N najboljih kandidata za rešenje i na osnovu njih smanjuje broj potencijalnih kandidata.
- **Spektralna konvolucija**: Određuje aminokiseline koje mogu da učestvuju u peptidu na osnovu eksperimentalnog spektra.
- **DeepNovo**: Metoda zasnovana na dubokom učenju koja omogućava sekvenciranje peptida bez oslanjanja na baze podataka.

3.1 Gruba sila (*Brute Force*)

Najosnovniji pristup sekvenciranju koji sistematski ispituje sve moguće kombinacije aminokiselina dok ne pronade sekvencu koja najbolje odgovara eksperimentalnom spektru [12, 14]. Iako jednostavan za implementaciju, ovaj pristup postaje neizvodiv za duže sekvence zbog eksponencijalnog rasta prostora pretrage.

Na primer, za peptid mase 579 Da, algoritam će generisati sve moguće kombinacije aminokiselina i proveriti da li njihova ukupna masa odgovara zadatoj masi.

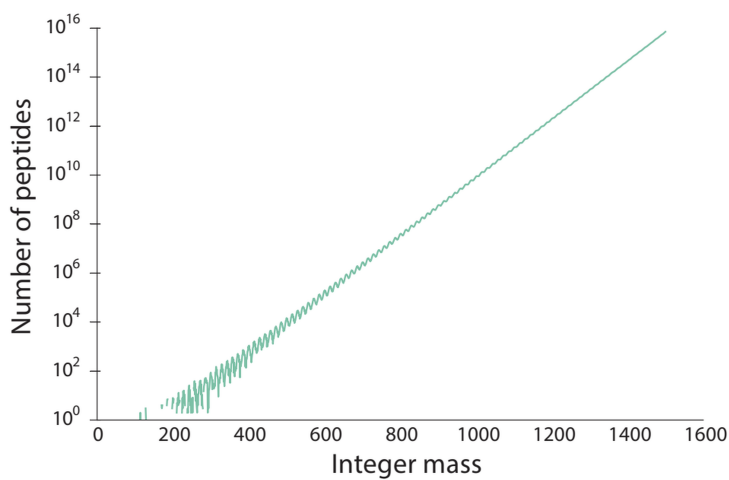
Kao što je prikazano u tabeli 3.1, mogu postojati različiti peptidi sa istom ukupnom masom (**FPAYT** i **QNWGS**), što dodatno komplikuje problem.

F	147 Da	Q	128 Da
P	97 Da	N	114 Da
A	71 Da	W	186 Da
Y	163 Da	G	57 Da
T	101 Da	S	94 Da
Ukupna masa: 579 Da		Ukupna masa: 579 Da	

Tabela 3.1: Poređenje aminokiselina i njihove mase

Da bi se utvrdilo koji od peptida je tačno rešenje, algoritam mora da generiše teorijski spektar za svaki kandidat peptid i uporedi ga sa eksperimentalnim spektrom. Ovo dodatno povećava računsku složenost algoritma, ali je neophodno za pronalaženje tačnog rešenja.

Na slici 3.1, možemo da vidimo koliko postoji različitih peptida za istu masu, samim tim možemo zaključiti da je izvršavanje algoritma grube sile veoma neefikasno.



Slika 3.1: Broj peptida koji imaju istu masu. Preuzeto iz [14].

Na osnovu prethodnog teksta definiše se pseudokod za algoritam grube sile koji se može videti kao algoritam 1.

Algoritam 1: Gruba sila

Funkcija GrubaSila(*eksperimentalniSpektar*)

```
    peptidi ← Lista sa praznim stringom
    rezultati ← Prazna lista
    ciljna_masa ← Poslednji element eksperimentalniSpektar
    while peptidi ≠ Prazno do
        prosireni ← Proširi(peptidi)
        kandidati ← Prazna lista
        foreach peptid ∈ prosireni do
            masa ← IzračunajMasu(peptid)
            if masa = ciljna_masa then
                if CikličniSpektar(peptid) = eksperimentalniSpektar then
                    Dodaj peptid u rezultati
                end
            else if masa < ciljna_masa then
                Dodaj peptid u kandidati
            end
        end
        peptidi ← kandidati
    end
    return rezultati
```

Objašnjenje pomoćnih funkcija:

- **Proširi(peptidi)** – sve preostale peptide proširuje sa svakom mogućom aminokiselinom i vraća proširenu listu.
- **IzračunajMasu(peptid)** – računa ukupnu masu peptida sabiranjem masa svih aminokiselina koje čine taj peptid.
- **CikličniSpektar(peptid)** – za peptid koji je potencijalno rešenje generiše se ciklični spektar koji se poredi sa zadatim eksperimentalnim spektrom.

3.2 Branch and Bound

Branch and Bound algoritam [12, 14] je optimizovana verzija algoritma grube sile koja koristi strategiju „podeli pa vladaj“ za efikasnije pretraživanje prostora

rešenja. Za razliku od algoritma grube sile koji ispituje sve moguće kombinacije, *Branch and Bound* algoritam inteligentno eliminiše delove prostora pretrage koji ne mogu sadržati optimalno rešenje.

U kontekstu sekvenciranja peptida, algoritam funkcioniše na sledeći način:

- **Grananje (*Branch*):** Algoritam gradi stablo pretrage gde svaki čvor predstavlja delimičnu sekvencu peptida. Svaki čvor se grana dodavanjem nove aminokiseline na postojeću sekvencu.
- **Ograničavanje (*Bound*):** Za svaki čvor, algoritam procenjuje da li taj put može dovesti do validnog rešenja. Ako masa peptida već premašuje ciljanu masu ili ako teorijski spektar delimične sekvence nije u skladu sa eksperimentalnim, ta grana se odseca i dalje se ne istražuje.
- **Optimizacija:** Algoritam može koristiti dodatne heuristike za procenu koje grane prvo istražiti, što dodatno ubrzava pronalaženje rešenja.

Prednosti *Branch and Bound* algoritma u odnosu na algoritam grube sile su značajne i njihovo poređenje može da se vidi u tabeli 3.2.

Algoritam grube sile	Branch and Bound
Istražuje sve moguće kombinacije	Inteligentno eliminiše neperspektivne grane
Eksponencijalna vremenska složenost	Značajno bolja vremenska složenost (u najgorem slučaju i dalje eksponencijalna, ali znatno brža)
Garantuje pronalaženje svih rešenja	I dalje garantuje pronalaženje svih rešenja
Neefikasan za duže peptide	Efikasniji za duže peptide

Tabela 3.2: Poređenje algoritma grube sile i Branch and Bound pristupa

Pseudokod za algoritam Branch and Bound može se videti kao algoritam 2.

Algoritam 2: Branch and Bound

Funkcija BranchAndBound(*eksperimentalniSpektar*)

```
    peptidi  $\leftarrow$  Lista sa praznim stringom
    rezultati  $\leftarrow$  Prazna lista
    ciljna_masa  $\leftarrow$  Poslednji element eksperimentalniSpektar
    while peptidi  $\neq$  Prazno do
        prosireni  $\leftarrow$  Proširi(peptidi)
        kandidati  $\leftarrow$  Prazna lista
        foreach peptid  $\in$  prosireni do
            masa  $\leftarrow$  IzračunajMasu(peptid)
            if masa = ciljna_masa then
                if CikličniSpektar(peptid) = eksperimentalniSpektar then
                    Dodaj peptid u rezultati
                end
            else if masa < ciljna_masa then
                if Konzistentan(peptid, eksperimentalniSpektar) then
                    Dodaj peptid u kandidati
                end
            end
        end
        peptidi  $\leftarrow$  kandidati
    end
    return rezultati
```

Objašnjenje pomoćnih funkcija:

- **Konzistentan**(*peptid*, *eksperimentalniSpektar*) – glavni korak odsecanja i smanjivanja prostora pretrage. Provera da li se svaka masa u okviru linearnog spektra peptida javlja i u okviru eksperimentalnog spektra. Ako neka masa postoji u okviru linearnog spektra a ne u okviru eksperimentalnog spektra to znaci da spektrum nije konzistentan, obrnuto ne mora da važi, jer se računa spektar parcijalnog peptida a ne još celog pa masa koja trenutno nije prisutna može da se pojavi.

3.3 Uporedna analiza prethodnih algoritama

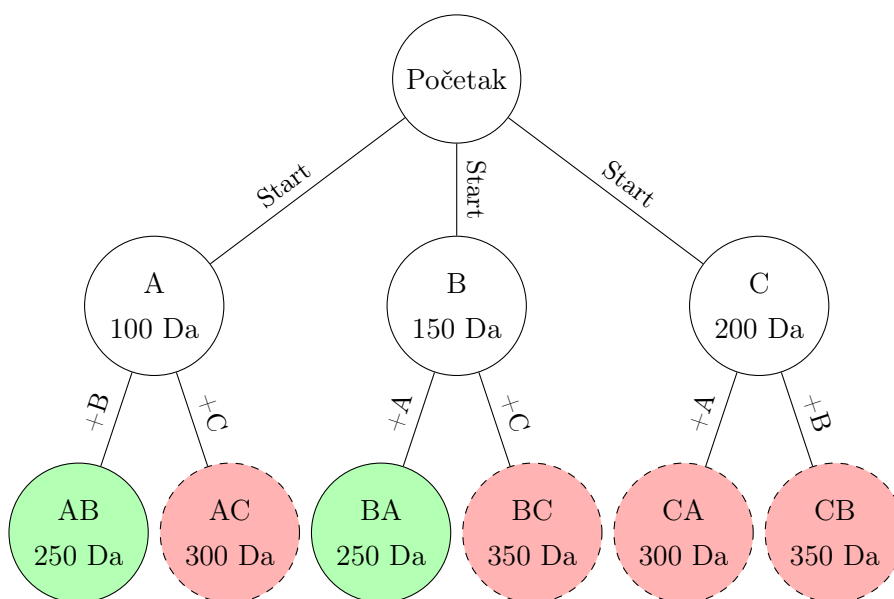
Posmatramo primer sa 3 aminokiseline:

- A (masa = 100 Da)
- B (masa = 150 Da)
- C (masa = 200 Da)

Tražimo sekvencu AB (ukupna masa = 250 Da). Poredimo algoritme grube sile i grananja i ograničavanja.

3.3.1 Algoritam grube sile

Ispituje sve kombinacije i odbacuje samo kada ukupna masa premaši traženu masu:

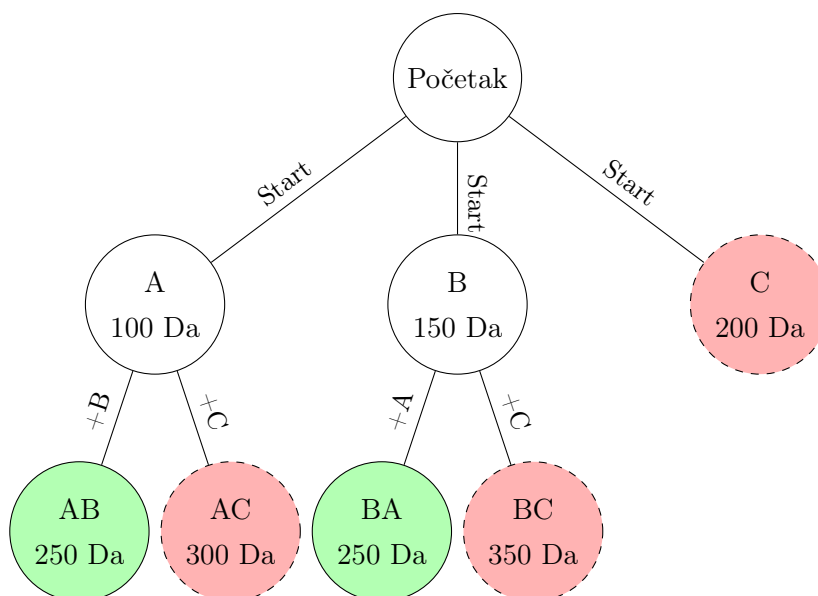


Objašnjenje:

- **Zeleno** - pronađena rešenja (AB, BA)
- **Crveno** - odbačene grane (masa > 250 Da)
- Ukupno evaluacija: 3 (prvi nivo) + 6 (drugi nivo) = 9 čvorova

3.3.2 Algoritam Branch and Bound

Odbacuje grane gde bilo koji deo sekvence nema masu koja postoji u eksperimentalnom spektru:



Objašnjenje:

- **Zeleno** - pronađena rešenja (AB, BA)
- **Crveno** - odbačene grane (C nema masu iz spektra)
- Ukupno evaluacija: 3 (prvi nivo) + 4 (drugi nivo) = 7 čvorova

3.3.3 Zaključak

Algoritam grananja i ograničavanja značajno smanjuje prostor pretrage eliminacijom neperspektivnih grana već u ranim fazama, dok gruba sila mora da ispita sve kombinacije. Na ovom primeru može se primetiti kako se odmah odbacuje cela **C** grana i svi njeni potomci. U ovom slučaju smanjuje se prostor pretrage za samo dva čvora, ali u stvarnosti postoji 20 aminokiselina i odsecanje čvorova i njihovih potomaka u samom startu predstavlja ogromno ubrzanje i poboljšanje u odnosu na algoritam grube sile.

3.4 Leaderboard algoritam

Leaderboard algoritam [12, 14] predstavlja optimizovani pristup za sekvenciranje peptida. Za razliku od sekvenciranja grubom silom koje zahteva tačno poklapanje između teorijskog spektra kandidata i eksperimentalnog spektra, ovaj algoritam je dizajniran da radi sa nedostajućim i lažnim masama tako što prati samo najbolje kandidate peptida umesto svih mogućnosti. *Leaderboard* algoritam održava listu najboljih kandidata tokom pretrage. U svakoj iteraciji, algoritam proširuje sekvence sa svim mogućim aminokiselinama i zadržava one kandidate čiji linearni spektar ima najveći broj poklapanja sa zadatim eksperimentalnim spektrom. Određuje se linearni spektar kandidata zato što peptidi još nisu do kraja formirani tako da se još ne zna kako bi oni izgledali ciklično.

3.4.1 Prednosti algoritma

- **Efikasnost:** Fokusira se samo na najperspektivnije kandidate, značajno smanjujući vreme izvršavanja.
- **Skalabilnost:** Efikasno radi sa peptidima različitih dužina bez eksponencijalnog rasta vremena.
- **Preciznost:** Održava visoku tačnost uprkos šumu u eksperimentalnim podacima.

3.4.2 Primene

- **Eksperimentalni podaci:** Idealan za analizu realnih podataka masene spektrometrije sa šumom. Pokazao se kao dobar algoritam koji može da pronađe rešenje i kada je broj lažnih ili nedostajućih masa iz spektra 10%, u slučaju kada se taj broj poveća na 25%, ovaj algoritam ne nalazi uvek tačno rešenje.
- **Nepotpuni podaci:** Kada tačno poklapanje nije moguće zbog nepotpunih ili netačnih podataka u spektru.
- **Vremenski kritične analize:** Kada je potrebna brza identifikacija peptida iz velikih skupova podataka.

Algoritam 3 prikazuje pseudokod *Leaderboard* algoritma.

Algoritam 3: Leaderboard sekvenciranje

Funkcija `Leaderboard(eksperimentalniSpektar)`

```
    peptidi ← Lista sa praznim stringom
    leader_peptid ← Prazan peptid
    najbolji_rezultat ← 0
    ciljna_masa ← Poslednji element eksperimentalniSpektar
    while peptidi ≠ Prazno do
        prosireni ← Proširi(peptidi)
        kandidati ← Prazna lista
        foreach peptid ∈ prosireni do
            masa ← IzračunajMasu(peptid)
            if masa = ciljna_masa then
                rezultat_kandidata ← CikličniScore(peptid,
                    eksperimentalniSpektar)
                if rezultat_kandidata > najbolji_rezultat then
                    leader_peptid ← peptid
                    najbolji_rezultat ← rezultat_kandidata
                end
            else if masa < ciljna_masa then
                Dodaj peptid u kandidati
            end
        end
        peptidi ← Trim(kandidati, eksperimentalniSpektar, N)
    end
    return leader_peptid
```

Objašnjenje pomoćnih funkcija:

- **Trim(kandidati, eksperimentalniSpektar, N)** – Jedna od glavnih funkcija. Ulaz u funkciju predstavljaju peptidi koji su kandidati za rešenje, eksperimentalni spektar kao i broj peptida koji ćemo vratiti iz funkcije odnosno najbolji kandidati za potencijalno rešenje. Bitno je izabrati dobro broj kandidata koji prolazi u dalju rundu. U slučaju da je taj broj previše mali rizujemo da previše agresivno odsečemo neke kandidate i da potencijalno izgubimo rešenje. U slučaju da je broj previše veliki čuvaćemo previše kandidata i samim tim povećati vreme izvršavanja algoritma. Generalno, dobra je praksa ako se traže peptidi sa manjom masom da se koristi manji broj kandidata koji nastavlja u

sledeću rundu a ako se masa poveća da se samim tim poveća i broj kandidata koji nastavlja u sledeću rundu. Funkcija **Trim** koristi funkciju **linearScore** koja računa broj poklapanja teorijskog spektra peptida sa eksperimentalnim spektrom. Ova funkcija se koristi kada se ceo peptid još ne zna i samim tim ne mogu da se kreiraju sve ciklične varijacije jer bi se dobile mase koje se možda ne bi dobile kada se peptid proširi aminokiselinama. Na kraju **Trim** kandidati se sortiraju opadajuće po linearnom skor i u sledeću rundu prolaze prvih **N** kandidata kao i svi kandidati koji imaju isti rezultat kao kandidat na poziciji **N**. Na ovaj način osiguravamo da sa sigurnošću svi dobri kandidati prođu u sledeću rundu.

- **CikličniScore(peptid, eksperimentalniSpektar)** – Računa broj poklapanja teorijskog spektra cikličnog peptida sa eksperimentalnim spektrom. Ova funkcija se koristi u slučaju da je masa peptida jednaka najvećoj teorijskoj masi jer je u tom slučaju formiran ceo peptid i mogu da se nađu svi podpeptidi.

3.5 Spektralna konvolucija

Spektralna konvolucija [12, 14] je tehnika koja se koristi za identifikaciju aminokiselina koje mogu biti prisutne u peptidu na osnovu eksperimentalnog spektra. Ova metoda analizira razlike između masa u spektru i identifikuje one koje odgovaraju masama aminokiselina.

Proces se sastoji iz dva glavna koraka:

- **Izračunavanje konvolucije:** Za svaki par masa u spektru, izračunava se njihova razlika. Ove razlike mogu odgovarati masama aminokiselina. Pravi se matrica konvolucije, koja predstavlja donju trougaonu matricu gde je prva vrsta i prva kolona eksperimentalni spektar a pozicije u matrici predstavljaju apsolutnu vrednost razlike elemenata sa tim indeksom iz spektra.
- **Identifikacija aminokiselina:** Najčešće razlike koje se pojavljuju u spektru verovatno odgovaraju aminokiselinama prisutnim u peptidu. Te aminokiseline se izdvajaju i koriste u *Leaderboard* algoritmu.

Glavna prednost ovog algoritma jeste to što u samom startu smanjuje skup aminokiselina koje mogu da učestvuju u građenju peptida, čime se algoritam dosta

ubrzava. Takođe, ovim se otvara mogućnost da se identifikuju nepoznate ili modifikovane aminokiseline. Još jedna od prednosti ovog algoritma jeste to što može da radi na eksperimentalnim spektrima koji imaju još više pogrešnih ili nedostajućih masa.

Algoritam 4 prikazuje pseudokod algoritma spektralne konvolucije.

Algoritam 4: Spektralna konvolucija

```

Funkcija SpektralnaKonvolucija(eksperimentalniSpektar)
    matrica_konvolucije  $\leftarrow$  Prazna lista
    n  $\leftarrow$  Broj elemenata u eksperimentalniSpektar
    for i  $\leftarrow$  0 to n - 1 do
        for j  $\leftarrow$  0 to i - 1 do
            masa  $\leftarrow$  S[i] - S[j]
            if  $57 \leq \textit{masa} \leq 200$  then
                Dodaj masa u konvolucija
            end
        end
    end
    broj_pojavljivanja  $\leftarrow$  Prazna mapa
    foreach masa u konvolucija do
        if masa  $\in$  broj_pojavljivanja then
            broj_pojavljivanja[masa]  $\leftarrow$  broj_pojavljivanja[masa] + 1
        end
        else
            broj_pojavljivanja[masa]  $\leftarrow$  1
        end
    end
    sortirane_frekvencije  $\leftarrow$  SortirajOpadajuće(broj_pojavljivanja)
    najcesce_mase  $\leftarrow$  Uzimamo S najčešćih masa iz
        sortirane_frekvencije
    leader_peptid  $\leftarrow$  LeaderboardSequencing(eksperimentalniSpektar,
        najcesce_mase)
    return leader_peptid

```

Objašnjenje pomoćnih funkcija:

- **SortirajOpadajuće**(*broj_pojavljivanja*) – U konkretnoj implementaciji matrica konvolucije je najlakše da bude lista da bi se što lakše iteriralo kroz

nju. Na osnovu matrice se formiraju frekvencije masa i sortira se opadajuće na osnovu broja pojavljivanja masa.

- **LeaderboardSequencing(eksperimentalniSpektar, najcesce_mase)** – Poziva se prethodno implementiran *Leaderboard* algoritam, samo se ne koriste sve aminokiseline nego se koristi smanjen skup aminokiselina za proširivanje i pravljenje novih peptida.

U tabeli 3.3 može da se vidi matrica konvolucije za eksperimentalni spektar:

0 114 128 129 242 243 257 371

	Mase (Da)						
	0	114	128	129	242	243	257
0	–	–	–	–	–	–	–
114	114	–	–	–	–	–	–
128	128	14	–	–	–	–	–
129	129	15	1	–	–	–	–
242	242	128	114	113	–	–	–
243	243	129	115	114	1	–	–
257	257	143	129	128	15	14	–
371	371	257	243	242	129	128	114

Tabela 3.3: Matrica konvolucije

Na osnovu tabele 3.3 možemo da vidimo koje su to mase koje se najviše puta ponavljaju. Vidimo da se mase 114, 128 i 129 pojavljuju 4 puta i njih ćemo sigurno koristiti u *Leaderboard* algoritmu, preostale mase koje bi se koristile zavise od broja **S**, koji nam govori koliko najčešćih masa ćemo uzeti.

Na osnovu tabele 1.1 možemo da vidimo da masa 114 odgovara aminokiselini sa skraćenicom **N**, da masa 128 odgovara **K** i **Q** aminokiselinama a da masa 129 odgovara aminokiselini **E**. Rešenja zadatog teorijskog spektra su peptidi **NQE** i **NKE** i njihove ciklične kombinacije. Možemo da primetimo da smo u samom startu suzili izbor aminokiselina sa 20 na samo 3 koje predstavljaju rešenje, čime smo mnogo ubrzali proces pronalaska peptida.

3.6 DeepNovo

Trenutne tehnike za sekvenciranje peptida (poput pretraživanja baze podataka i *de novo* sekvenciranja) mogu imati poteškoća u radu sa novim, složenim ili nepotpunim podacima [17]. Pristup pretraživanja baze podataka oslanja se na poređenje eksperimentalnih podataka sa bazom podataka poznatih proteinskih sekvenci, ali neki od problema u ovom pristupu su sledeći:

- **Nepoznati proteini** - Novi proteini koji nikada ranije nisu viđeni neće se podudarati ni sa čim u bazi podataka, što dovodi do nemogućnosti identifikacije.
- **Nedostajući podaci** - Podaci generisani iz eksperimenata masene spektrometrije mogu biti pogrešni i nepotpuni, što otežava pouzdano poređenje.

Pristup *de novo* sekvenciranja pokušava izgraditi sekvencu peptida iz početka, bez oslanjanja na bazu podataka, ali je često manje precizan i računski skup [17]. *DeepNovo* kombinuje prednosti oba pristupa koristeći duboko učenje za poboljšanje tačnosti *de novo* sekvenciranja.

3.6.1 Računska složenost

De novo sekvenciranje, koje pokušava rekonstruisati sekvencu peptida bez oslanjanja na bazu podataka, suočava se sa značajnim računskim izazovima:

- Eksponencijalni rast prostora pretrage sa povećanjem dužine peptida
- Potreba za složenim algoritmima za interpretaciju spektralnih podataka
- Teškoće u razlikovanju izobaričnih aminokiselina (aminokiseline sa istom ili vrlo sličnom masom)

3.6.2 Tehnike zasnovane na De Novo sekvenciranju

Pored **DeepNovo** tehnike koja će biti opisana u ovom radu, postoje i još neke tehnike zasnovane na *De Novo* principu:

- **PEAKS** [16] - koristi direktne aciklične grafove i određuje najbolji rezultat
- **Novor** [13] - koristi klasifikatore mašinskog učenja da odredi sekvencu aminokiselina sa najvećom verovatnoćom

- **PepNovo** [10] - koristi modelovanje verovatnoća pomoću grafova

DeepNovo je dizajniran da prevazilazi računske izazove koristeći moć dubokog a rezultati će pokazati da je bolji i od drugih metoda koje su zasnovane na *de novo* principu.

3.6.3 Opšti pregled

DeepNovo [17] je metoda zasnovana na dubokom učenju koja poboljšava sekvenciranje peptida koristeći algoritam za predviđanje sekvenci aminokiselina iz podataka generisanih masenom spektrometrijom.

Osnovna ideja je da model dubokog učenja može naučiti obrasce u podacima masene spektrometrije i davati predviđanja o sekvenci peptida bez potrebe za oslanjanjem na referentnu bazu podataka.

Ova inovativna metoda pokazuje značajno poboljšanje u tačnosti sekvenciranja, posebno u slučajevima kada su peptidne sekvence nove i ne podudaraju se ni sa jednom poznatom proteinskom bazom podataka.

Proces treniranja

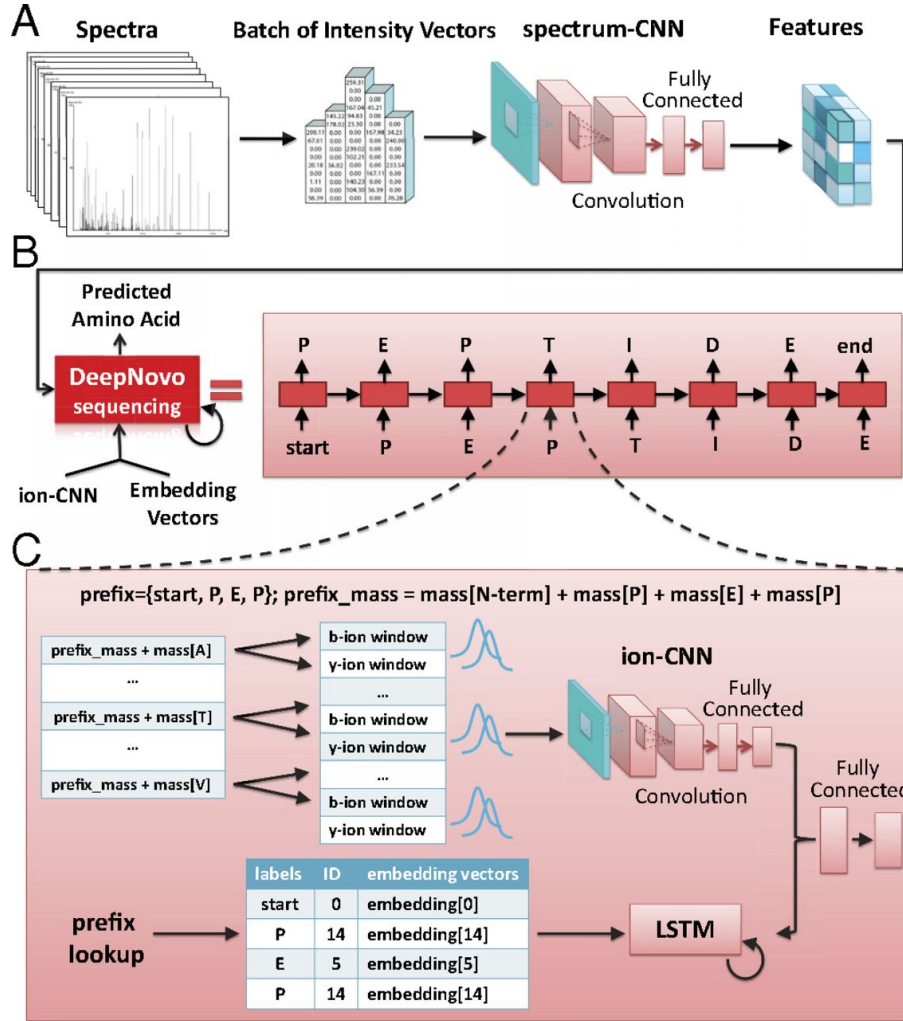
DeepNovo se trenira na velikom skupu podataka poznatih sekvenci peptida i njihovih odgovarajućih podataka masene spektrometrije. Model uči da preslikava eksperimentalne podatke na sekvence peptida kroz proces nadziranog učenja, gde se optimizuju parametri mreže da bi se minimizirala razlika između predviđenih i stvarnih sekvenci.

Proces treniranja obuhvata sledeće ključne faze:

1. Prikupljanje velikog skupa podataka poznatih peptidnih sekvenci i njihovih spektara
2. Pretprocesiranje spektralnih podataka za normalizaciju i uklanjanje šuma
3. Treniranje neuronske mreže da predviđa aminokiseline na osnovu spektralnih karakteristika
4. Optimizacija modela korišćenjem tehnika kao što su regularizacija i rano zaustavljanje
5. Validacija modela na nezavisnom skupu podataka za procenu performansi

3.6.4 Arhitektura neuronske mreže

DeepNovo koristi hibridnu arhitekturu koja kombinuje konvolucione i rekurentne neuronske mreže i može se videti na slici 3.2.



Slika 3.2: Arhitektura DeepNovo pristupa, preuzeto sa [17]

Konvolucione neuronske mreže (CNN)

CNN se koristi za otkrivanje značajnih šablona u ulaznim podacima. Veoma je efikasna mreža i koristi tehniku *sliding window* i procesira male lokalne regione koristeći filtere. U slučaju **DeepNovo** tehnike, konvoluciona mreža se sastoji od 3 konvoluciona sloja i koristi *ReLU* aktivacionu funkciju. Ova mreža je trenirana da prepozna lokalne šablone, različite tipove jona i da pretvori sirove podatke u reprezentaciju svojstava ulaznih podataka.

U ovom modelu su korišćene 2 **CNN** mreže:

- **Spektralna CNN**: Spektar dobijen masenom spektrometrijom konvertuje u vektor fiksne dužine (koji obično ima od 50 do 100 hiljada elemenata) i dobijeni vektor se prosleđuje ovoj mreži. Ovime se uče šabloni nad svim podacima spektrometrije i rezultat ove mreže se dalje koristi u rekurentnoj mreži. Ovo predstavlja značajan deo arhitekture jer može dosta da poboljša preciznost sekvenciranja i može da nauči šablone u spektru čak i ako su neki peak-ovi pomereni u spektru zbog šuma.
- **Jonska CNN**: Ova mreža se koristi tokom odabira sledeće aminokiseline u peptidu i ona služi da za mali region spektralnih podataka izvuče najbitnije informacije. Gleda da li za predviđenu aminokiselinu postoje očekivani fragmenti jona. Prilikom svakog koraka predviđanja sledeće aminokiseline **DeepNovo** generiše teorijski spektar fragmenata uz pomoć prefiksni masa. Za svaki jon izvlači se mali prozor iz spektra oko tog jona i na kraju se dobije više različitih ulaza u mrežu. Ovaj deo modela je bitan u slučaju da su podaci šumoviti ili da neki vrh spektra nedostaje.

Rekurentne neuronske mreže (RNN)

Ove mreže su dizajnirane za predviđanje sekvenci, gde izlaz zavisi ne samo od trenutne tačke podataka (podaci masenog spektra) već i od prethodnih tačaka podataka. U kontekstu sekvenciranja peptida, **RNN** može naučiti kako jedna aminokiselina utiče na sledeću u sekvenci, što je ključno za tačno predviđanje.

DeepNovo koristi posebnu vrstu RNN-a koja se zove *Long Short-Term Memory (LSTM)*. Ključna prednost **LSTM** mreža je ta što bolje prati duže zavisnosti, konkretno peptidi mogu da budu različitih dužina a ova mreža pamti i odnose koji su udaljeni, odnosno početak sekvence može da utiče na predviđanje neke kasnije aminokiseline.

Ova mreža se sastoji od 1 sloja **LSTM**-a i radi tako što dodaje jednu po jednu aminokiselinu sve dok ne stigne do kraja peptida. U svakom koraku **LSTM** mreža gleda:

- Šta je mreža naučila do sada - trenutno stanje
- Sledeća aminokiselina koja je kandidat

- Svojstva spektra koja su dobijana od konvolucione mreže

Na osnovu ovoga, mreža određuje koja je verovatnoća da je trenutna aminokiselina zapravo nalazi na datoj poziciji u peptidu.

Kao izlaz iz ove mreže koristi se *softmax* projekcija i ona određuje za svaku aminokiselinu koja je verovatnoća da se ona nalazi na sledećoj poziciji u sekvenci. Dodatno, koristi se *beam search*, odnosno ne bira se samo aminokiselina sa najvećom verovatnoćom nego se čuva više kandidata koji imaju veću verovatnoću. Ovim postupkom se povećava preciznost i gledaju se alternativna rešenja. Na kraju se sekvence rangiraju po rezultatu koliko se poklapaju sa traženim spektrom i koliko imaju grešaka i bira najbolja moguća.

3.6.5 Rezultati i evaluacija

DeepNovo metoda nadmašuje tradicionalne metode, posebno u slučajevima kada su sekvence peptida nove i ne podudaraju se ni sa jednom poznatom proteinskom bazom podataka.

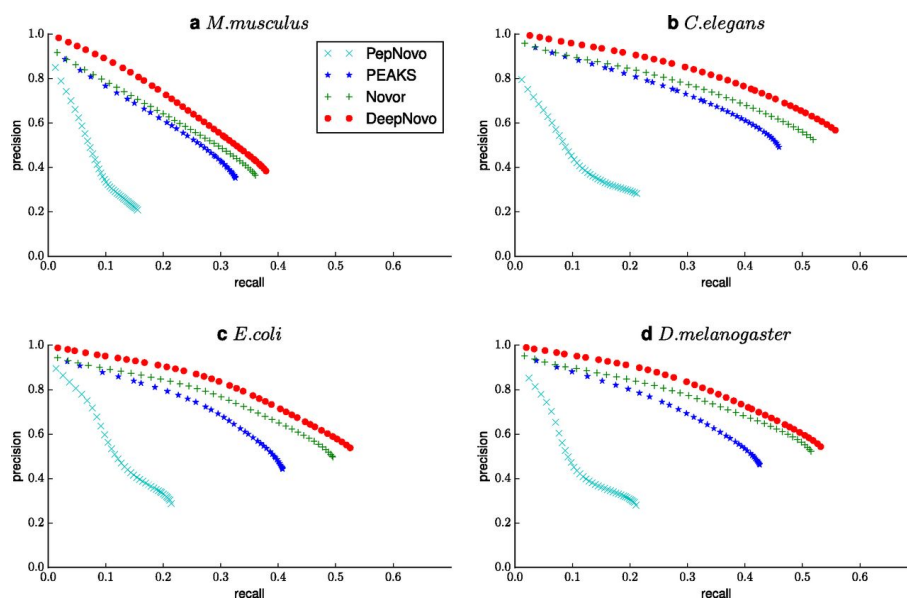
Eksperimenti su pokazali značajno poboljšanje u tačnosti identifikacije aminokiselina, posebno za složene peptide.

Za potrebe testiranja naučnici su koristili podatke različitih vrsta. Na slici 3.3 su prikazani rezultati poređenja više algoritama. Da bi se merila preciznost rešenja poredena je prava sekvenca aminokiselina sa onom koja je dobijena na osnovu spektra. Takođe, koristile su se različite metrike:

- **Preciznost (eng. precision)** - koji predstavlja broj peptida koji je generisao algoritam koji su zapravo tačni
- **Odziv (eng. recall)** - koji broj stvarnih peptida je uspešno pronađen od strane modela
- **AUC-PR** - koliko dobro model balansira preciznost i odziv, što veća vrednost bolje će biti i preformanse

Može se primetiti da je **DeepNovo** model na svakom od datih skupa podataka imao bolje rezultate. **DeepNovo** je imao i veću preciznost u traženju peptida kao i veći odziv, samim tim i odnos *AUC-PR* krive je bolji nego kod konkurenata.

DeepNovo je uspešno primenjen u nekoliko realnih scenarija:



Slika 3.3: Poređenje performansi DeepNovo sa drugim algoritmima, preuzeto sa [17]

- **Identifikacija novih antimikrobnih peptida:** Otkrivanje potencijalnih kandidata za nove antibiotike
- **Analiza post-translacionih modifikacija:** Detekcija peptida sa složenim modifikacijama koje su se desile nakon njegovog kreiranja
- **Univerzalna primena:** Uspešna implementacija na različitim vrstama i organizmima

Glava 4

Elektronska platforma

U ovom poglavlju će biti opisana elektronska platforma koja je napravljena kao deo ovog rada. Objasniće se njeno pokretanje i korišćenje kao i tehnologije koje su korišćene prilikom pravljenja platforme.

Frontend aplikacije je pisan u programskom jeziku **TypeScript** [9] uz korišćenje **Node 18** [7] i **Next.js framework**-a [6]. *Backend* aplikacije je implementiran u programskom jeziku **Python 3.12** [8] uz korišćenje **Django framework**-a [1]. Izvorni kod aplikacije se nalazi na *GitHub*-u, u javnom repozitorijumu [4].

4.1 Pokretanje aplikacije

Pokretanje aplikacije može da se odradi na nekoliko načina:

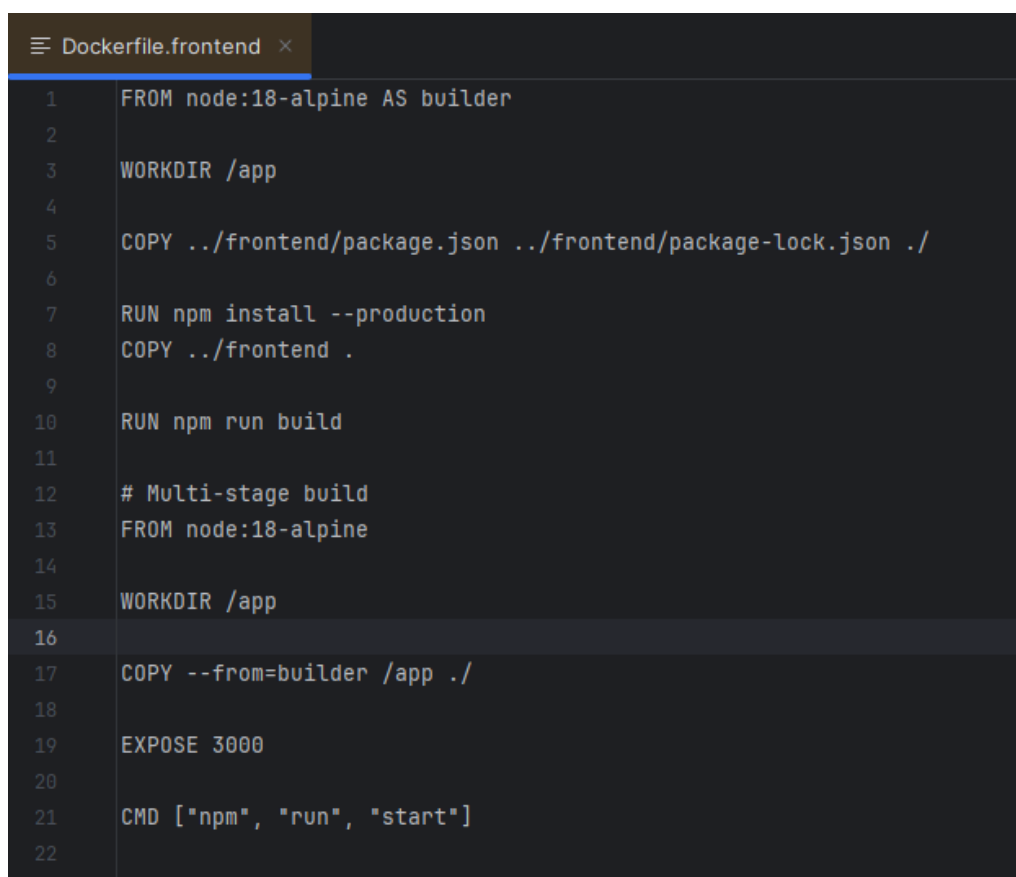
- **Docker Compose** [2] - najjednostavniji način pokretanja aplikacije uz korišćenje *Docker* alata [3]
- **Direktno pokretanje komponenti** - pokretanje posebno *Frontend* i posebno *Backend* komponenti
- **Google Cloud Run (GCR)** [5] - ova aplikacija je dostupna za korišćenje preko *Google Cloud Run* platforme, pa će se samim tim objasniti i kako odraditi *deploy* aplikacije na **GCR**

4.1.1 Docker Compose konfiguracija

Za potrebe što lakšeg pokretanja aplikacije korišćen je *open source* alat *Docker* i *Docker Compose*. *Docker Compose* nam pruža jednostavan način da aplikaciju

pokrenemo sa svim definisanim bibliotekama, promenljivama okruženja bez potrebe da se bilo šta dodatno namesti. Ovo je posebno bitno u slučaju kada se aplikacija sastoji iz više komponenti pa je potrebno da se pokrene više kontejnera kao što je ovde slučaj. Unutar **docker** foldera nalaze *docker* i *docker-compose* fajlovi.

Što se tiče klijentskog dela aplikacije njegovo pokretanje je definisano u *Dockerfile.frontend* datoteci i njegov sadržaj može da se vidi na slici 4.1.



```
1 FROM node:18-alpine AS builder
2
3 WORKDIR /app
4
5 COPY ../frontend/package.json ../frontend/package-lock.json ./
6
7 RUN npm install --production
8 COPY ../frontend .
9
10 RUN npm run build
11
12 # Multi-stage build
13 FROM node:18-alpine
14
15 WORKDIR /app
16
17 COPY --from=builder /app ./
18
19 EXPOSE 3000
20
21 CMD ["npm", "run", "start"]
22
```

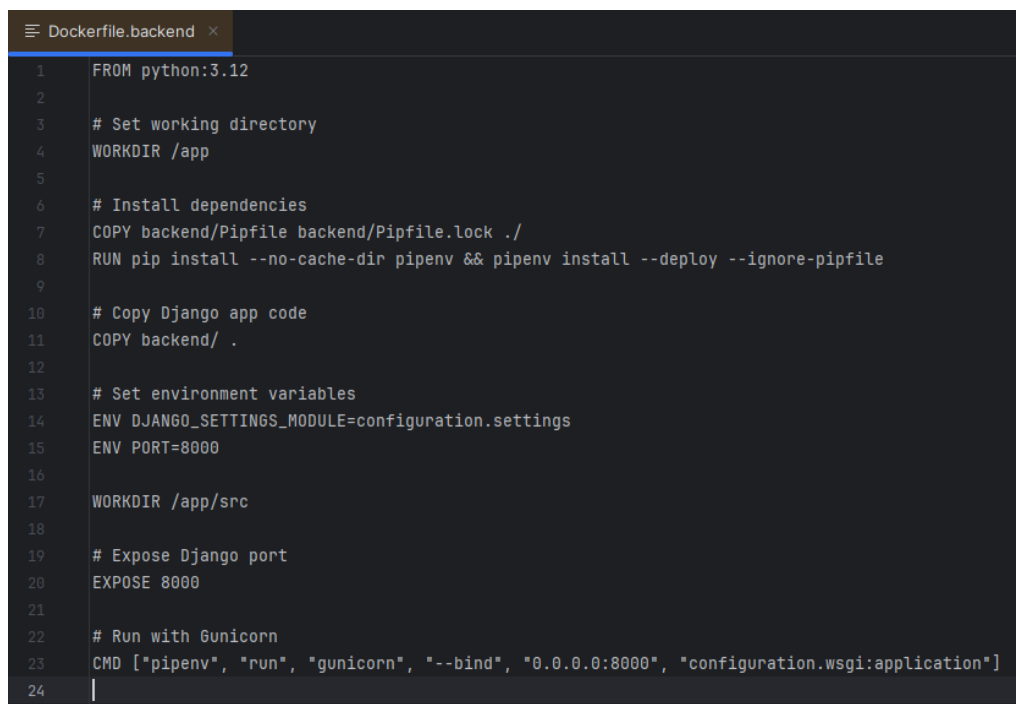
Slika 4.1: *Dockerfile* za klijentski deo aplikacije

Može da se primeti da se koristi *multi-stage build* koji služi da odvoji deo za instaliranje i kompilaciju fajlove od dela koji će pokrenuti aplikaciju. Ovo doprinosi tome da završa slika bude što manja.

Pokretanje serverskog dela aplikacije definisano je u *Dockerfile.backend* datoteci i njegov sadržaj može da se vidi na slici 4.2.

Može da se primeti da se koristi *Gunicorn* koji predstavlja *WSGI HTTP* server za produkciona okruženja.

Sadržaj glavne *docker-compose.yml* datoteke može da se vidi na slici 4.3.

A screenshot of a code editor showing a Dockerfile for a Django backend application. The file is named 'Dockerfile.backend' and contains 24 lines of code. The code starts with 'FROM python:3.12', sets the working directory to '/app', installs dependencies using 'pipenv', copies the Django app code, sets environment variables for Django settings and port, and finally runs the application using 'gunicorn' on port 8000.

```
1 FROM python:3.12
2
3 # Set working directory
4 WORKDIR /app
5
6 # Install dependencies
7 COPY backend/Pipfile backend/Pipfile.lock ./
8 RUN pip install --no-cache-dir pipenv && pipenv install --deploy --ignore-pipfile
9
10 # Copy Django app code
11 COPY backend/ .
12
13 # Set environment variables
14 ENV DJANGO_SETTINGS_MODULE=configuration.settings
15 ENV PORT=8000
16
17 WORKDIR /app/src
18
19 # Expose Django port
20 EXPOSE 8000
21
22 # Run with Gunicorn
23 CMD ["pipenv", "run", "gunicorn", "--bind", "0.0.0.0:8000", "configuration.wsgi:application"]
24
```

Slika 4.2: *Dockerfile* za serverski deo aplikacije

Da bi se projekat pokrenuo potrebno je pozicionirati se u **/docker** direktorijum i pokrenuti sledeću komandu:

```
docker-compose up -d --build
```

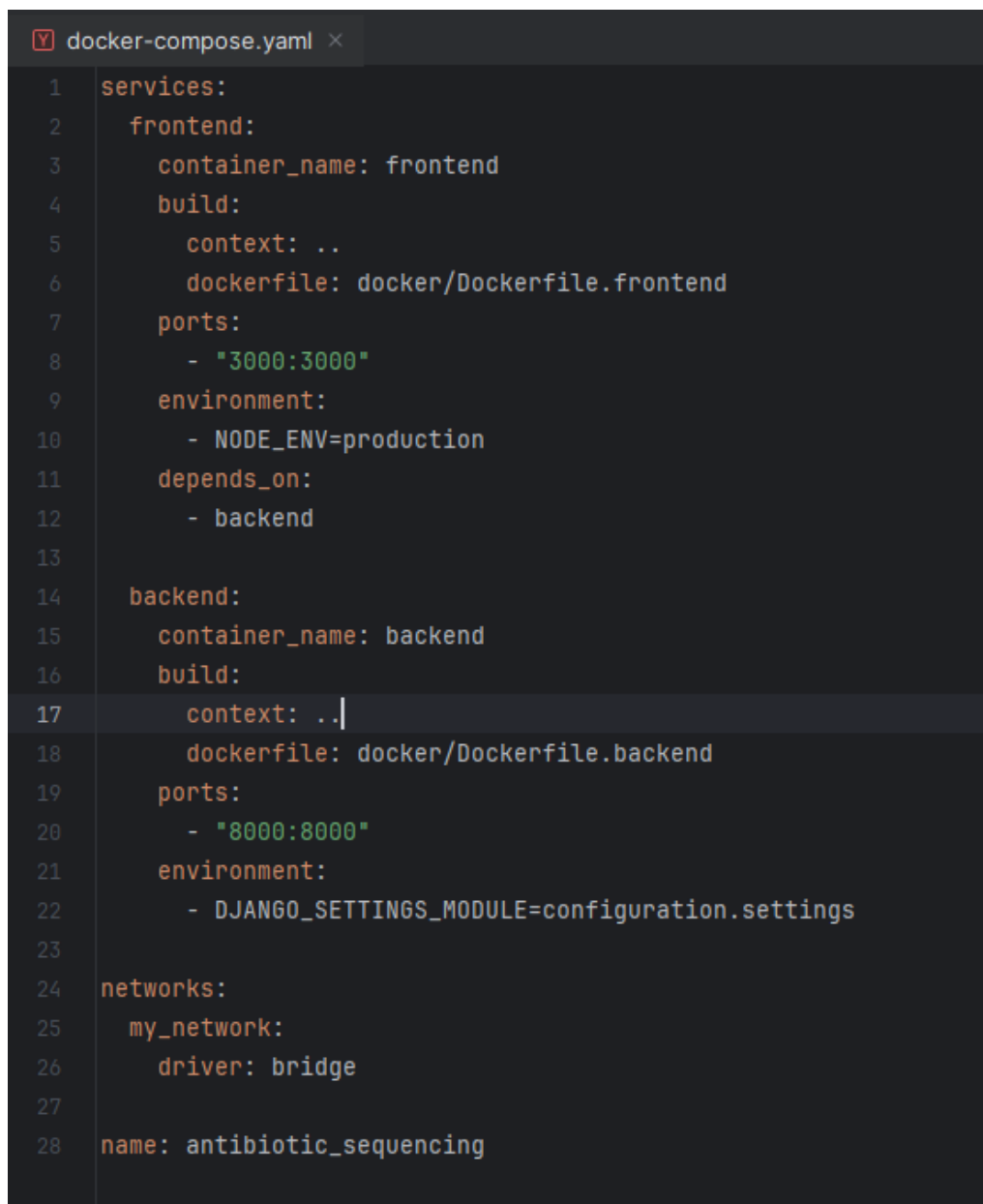
Ova komanda pokreće dva kontejnera. *Frontend* delu aplikacije može da se pristupi tako što se u veb pregladaču otvori <http://localhost:3000>, dok se *backend* deo aplikacije nalazi na <http://localhost:8000>.

4.1.2 Direktno pokretanje komponenti

Da bi se direktno pokrenuo *Frontend* aplikacije potrebno je imati instaliran **Node 18** kao i **npm 10** paket menadžer. Potrebno je pozicionirati se u **/frontend** direktorijum i pokrenuti naredne komande:

```
npm install
npm run build
npm run start
```

Nakon pokretanja ovih komandi klijentskom delu aplikacije može da se pristupi iz veb pregledača na adresi <http://localhost:3000>.



```
1 services:
2   frontend:
3     container_name: frontend
4     build:
5       context: ..
6       dockerfile: docker/Dockerfile.frontend
7     ports:
8       - "3000:3000"
9     environment:
10      - NODE_ENV=production
11     depends_on:
12      - backend
13
14   backend:
15     container_name: backend
16     build:
17       context: ..|
18       dockerfile: docker/Dockerfile.backend
19     ports:
20       - "8000:8000"
21     environment:
22      - DJANGO_SETTINGS_MODULE=configuration.settings
23
24 networks:
25   my_network:
26     driver: bridge
27
28 name: antibiotic_sequencing
```

Slika 4.3: *Docker Compose* fajl koji pokreće definisane *Docker* fajlove

Da bi se direktno pokrenuo *Backend* aplikacije potrebno je imati instaliran **Python 3.12** kao i **pip** paket menadžer. Potrebno je pozicionirati se u **/backend** direktorijum i izvršiti naredne komande:

```
python -m venv venv
venv\Scripts\activate
pip install pipenv
```

```
pipenv install -d
cd src
python manage.py runserver
```

Pokretanjem ovih komandi kreiraćemo virtuelno okruženje u kom će se instalirati sve zavisnosti ove aplikacije. Za praćenje verzije korišćenih biblioteka korišćen je *Pipfile* i zato mora da se instalira i *pipenv*. Nakon pokretanja serverskom delu aplikacije mogu se slati zahtevi na adresu *http://localhost:8000*.

4.1.3 Google Cloud Run implementacija

Aplikacija je trenutno *deploy*-ovana na Google Cloud Run i može joj se pristupiti preko *https://antibiotic-sequencing-304513663933.us-central1.run.app/*. Koristi se besplatna verzija koja se ne naplaćuje a pored toga *Google Cloud Run* pruža sledeće pogodnosti:

- **Skalabilnost:** Automatsko skaliranje u zavisnosti od opterećenja
- **Integracija:** Potpuna podrška za *Docker* kontejnere
- **Bezbednost:** Ugrađena zaštita *DDoS* napada
- **Monitoring:** Integrisani *Cloud Monitoring* alati

Postoji *Google Cloud CLI* alat koji može da se instalira i da se iz terminala pokreću odgovarajuće komande. Proces se sastoji iz sledećih koraka i potrebno je da se ovi koraci odrade i za klijentsku i za serversku komponentu:

1. Pravljenje projekta na *Google Cloud* platformi čiji će se *PROJECT_ID* dalje koristiti
2. Povezivanje *Google Cloud CLI* alata sa *Google* nalogom:

```
gcloud auth login
```

3. Definisanje sa kojim projektom želimo da radimo:

```
gcloud config set project <PROJECT_ID>
```

4. Pravljenje *Docker* image-a:

```
docker build -t gcr.io/<PROJECT_ID>/fe:v1.0
-f docker/Dockerfile.frontend .
```

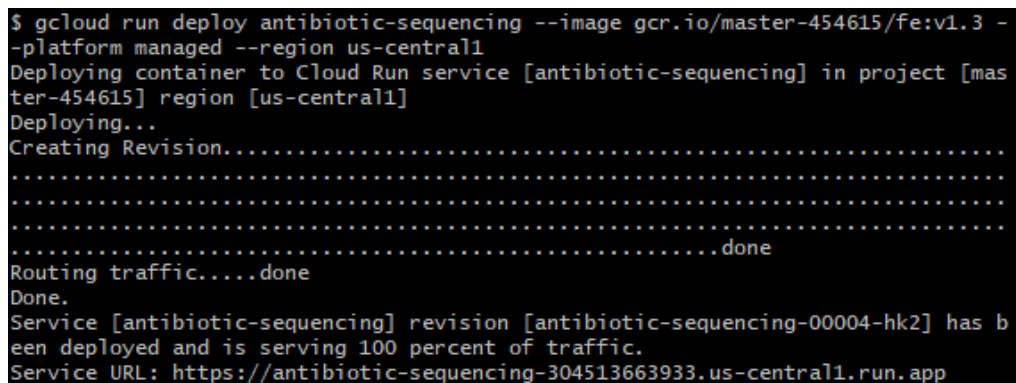
5. Kačenje slike na njihov repozitorijum:

```
docker push -t gcr.io/<PROJECT_ID>/fe:v1.0
```

6. Deploy servisa:

```
gcloud run deploy antibiotic-sequencing
--image gcr.io/<PROJECT_ID>/fe:v1.0
--platform managed
--region us-central1
--allow-unauthenticated
--port 3000
```

Nakon toga u terminalu se dobija *URL* na kom može da se pristupi podignutom servisu što se može videti na slici 4.4. Takođe u toj situaciji frontend komponenti mora u */.env* fajlu da se promeni *URL* koji vodi do backend komponente.



```
$ gcloud run deploy antibiotic-sequencing --image gcr.io/master-454615/fe:v1.3 -
--platform managed --region us-central1
Deploying container to Cloud Run service [antibiotic-sequencing] in project [mas
ter-454615] region [us-central1]
Deploying...
Creating Revision.....done
.....done
Routing traffic.....done
Done.
Service [antibiotic-sequencing] revision [antibiotic-sequencing-00004-hk2] has b
een deployed and is serving 100 percent of traffic.
Service URL: https://antibiotic-sequencing-304513663933.us-central1.run.app
```

Slika 4.4: Dobijeni *URL* nakon što se aplikacija podigne na *Google Cloud Run* servisu

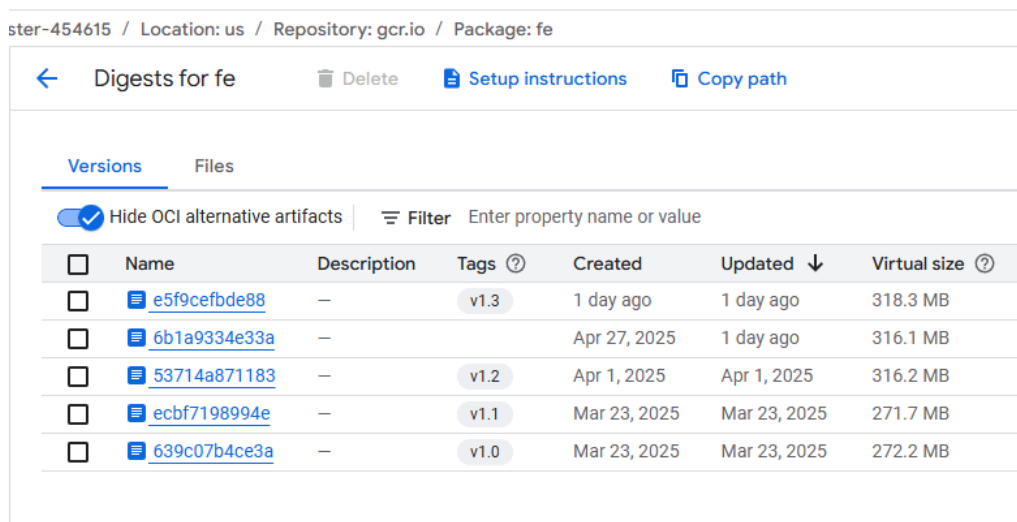
U slučaju da je potrebno okačiti novu verziju aplikacije, prate se isti koraci za pravljenje i kačenje slike dok se komanda za *deploy* razlikuje jer se navodi kako se zove komponenta koja se ažurira:

```
gcloud run deploy antibiotic-sequencing
--image gcr.io/<PROJECT_ID>/fe:v1.1
--platform managed
```



```
--region us-central1
```

Ako želimo da pristupimo *Docker* slikama koje smo okačili potrebno je otvoriti <https://console.cloud.google.com/artifacts>, slika 4.5.



ster-454615 / Location: us / Repository: gcr.io / Package: fe

← Digests for fe Delete Setup instructions Copy path

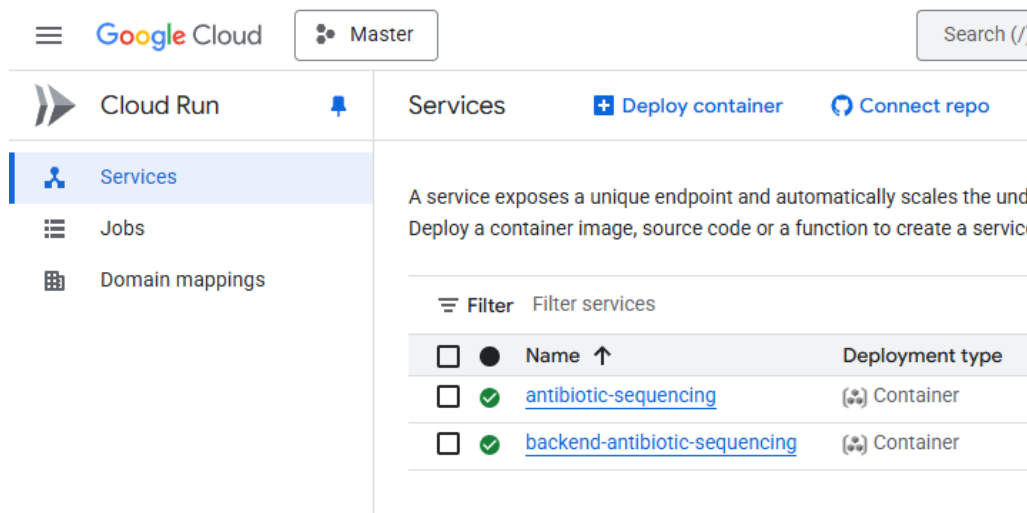
Versions Files

☒ Hide OCI alternative artifacts Filter Enter property name or value

<input type="checkbox"/>	Name	Description	Tags ?	Created	Updated ↓	Virtual size ?
<input type="checkbox"/>	e5f9cefbde88	—	v1.3	1 day ago	1 day ago	318.3 MB
<input type="checkbox"/>	6b1a9334e33a	—		Apr 27, 2025	1 day ago	316.1 MB
<input type="checkbox"/>	53714a871183	—	v1.2	Apr 1, 2025	Apr 1, 2025	316.2 MB
<input type="checkbox"/>	ecbf7198994e	—	v1.1	Mar 23, 2025	Mar 23, 2025	271.7 MB
<input type="checkbox"/>	639c07b4ce3a	—	v1.0	Mar 23, 2025	Mar 23, 2025	272.2 MB

Slika 4.5: *Docker* slike koje su okačene na *Google Container Registry*

Podignutim servisima može se pristupiti preko <https://console.cloud.google.com/run>, slika 4.6.



Google Cloud Master Search (/)

Cloud Run Services Deploy container Connect repo

Services

A service exposes a unique endpoint and automatically scales the und
Deploy a container image, source code or a function to create a service

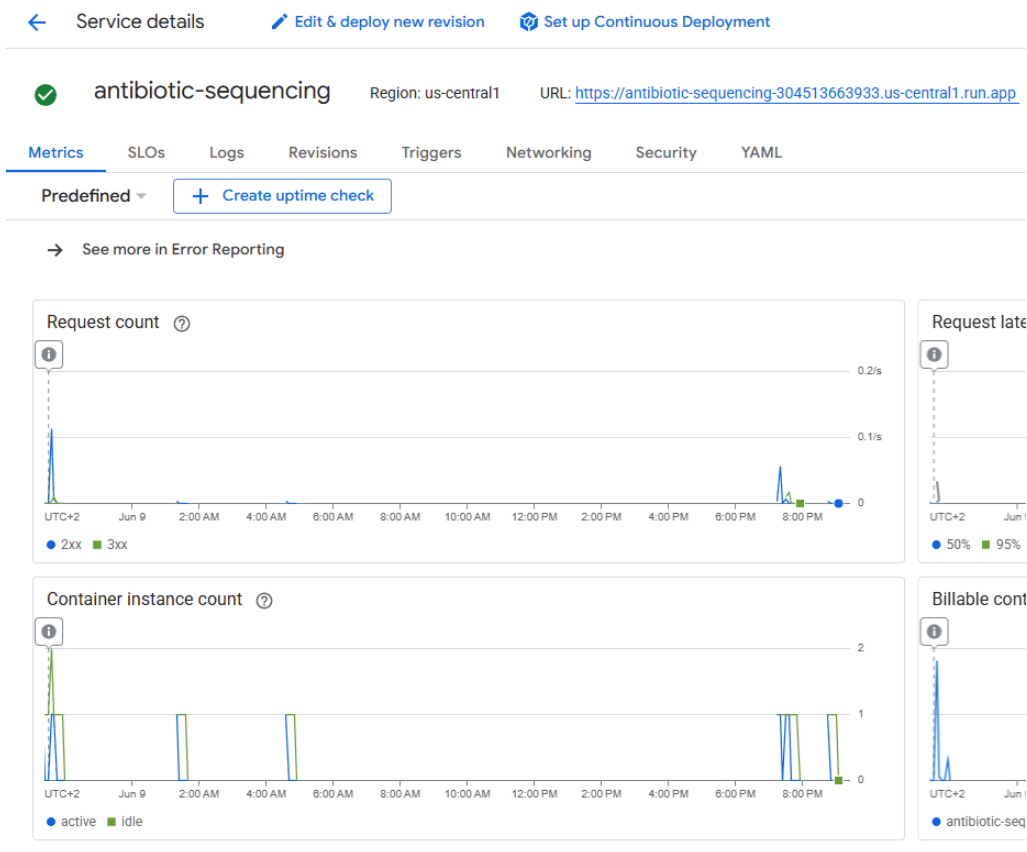
Filter Filter services

<input type="checkbox"/>	●	Name ↑	Deployment type
<input type="checkbox"/>	✓	antibiotic-sequencing	(🔗) Container
<input type="checkbox"/>	✓	backend-antibiotic-sequencing	(🔗) Container

Slika 4.6: Pristupi servisima koji su podignuti na *Google Cloud Run* platformi

Ako se klikne na neki od servisa pristupiće se brojnim metrikama koje su nam dostupne, 4.7. Takođe, tu možemo da vidimo i koji je *URL* našeg servisa kao i da

vidimo koji je status i da li su se desile neke greške.



Slika 4.7: Detalji i metrike koje su nam dostupne kada pristupimo nekom servisu

4.2 Funkcionalnosti platforme

4.2.1 Početna stranica

Prilikom otvaranja aplikacije otvoriće se početna stranica koja može da se vidi na slici 4.8. Na vrhu stranice nalazi se navigacioni meni kojim može da se prelazi sa stranice na stranicu. Takođe, u gornjem desnom uglu nalazi se ikonica koja vodi ka *Github* repozitorijumu gde može da se nađe izvorni kod ove aplikacije. Klikom na dugme *Istraži algoritme* odlazi se na stranicu *Uvod*, gde mogu da se nađu teorijska objašnjenja pojma koja su potrebna za razumevanje algoritama.

4.2.2 Uvodna stranica

Ova stranica služi da predstavi teorijske osnove i pojmove koji su potrebni za dalje razumevanje algoritama. Na slici 4.9 može da se vidi deo ove stranice. Klikom na bilo koju sliku sa ove stranice ona će se otvoriti i omogućiti jasniji prikaz iste.

Na dnu stranice stranice, koja je prikazana na slici 4.10, takođe postoji meni koji vodi ka algoritmima koji su obrađeni u sklopu ovde aplikacije.



Slika 4.8: Početna stranica kada se otvori aplikacija



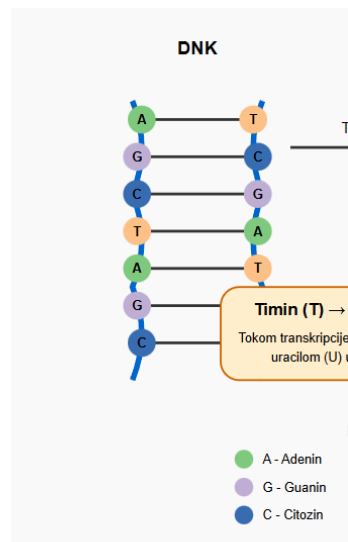
Uvod u sekvenciranje antibiotika

Proces sekvenciranja antibiotika je fundamentalan u razumevanju kako su ovi molekuli proizvedeni od strane bakterija i kako se oni mogu sintetizovati ili modifikovani za primene u medicini. Antibiotici su često peptidi - kratki proteini odnosno kratak niz aminokiselina, ali mnogo antibiotika, uglavnom neribozomalni peptidi (*non-ribosomal peptides - NRPs*), ne prati standardna pravila za sintezu proteina čime se otežava njihovo sekvenciranje [1].

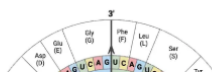
DNK sadrži recept za kreiranje proteina. Odnosno, sastoji se od gena koji mogu biti uključeni i tada će se na osnovu njih kreirati proteini ili isključeni kada se oni neće koristiti za kreiranje proteina. Isključenost ili uključenost nekog gena zavisi od toga da li je potrebno da se kreira neki protein ili nije potrebno (npr. fotosinteza kod biljaka koja se obavlja samo preko dana).

Tradicionalno, proteini prate **Centralnu Dogmu Molekularne biologije**, koja kaže da se DNK prvo prepisuje u RNK - slika 1, a zatim se RNK prevodi u protein. Na slici jedan se može se primetiti da se DNK sastoji od 2 lanca koja su komplementarna. Enzim *RNK polimeraza* se kači na početak gena i kreće kroz gene gde razdvaja lanac i stvara prostor za prepisivanje DNK u RNK čime se dobija RNK.

Prilikom prevođenja RNK u protein potrebno je na osnovu nukleotida odrediti koja je aminokiselina u pitanju. Organela ribozom je zadužena da odradi ovaj posao i pošto je potrebno na osnovu nukleotidne sekvence uniformno odrediti koja je aminokiselina u pitanju uzima se sekvenca od 3 nukleotida takođe poznata kao kodon. Pošto je uzeta sekvenca od 3 nukleotida ovo nam daje 64 različita kodona koja treba da se prevedu u 20 aminokiselina, da smo uzeli sekvenca od 2 nukleotida dobili bismo 16 različitih kombinacija čime ne bismo mogli da dobijemo sve aminokiseline. Na slici 2 može se videti kako se kodoni prevode u odgovarajuće aminokiseline. Postoje start i stop kodoni koji određuju početak odnosno kraj sekvence koja se prevodi u protein.

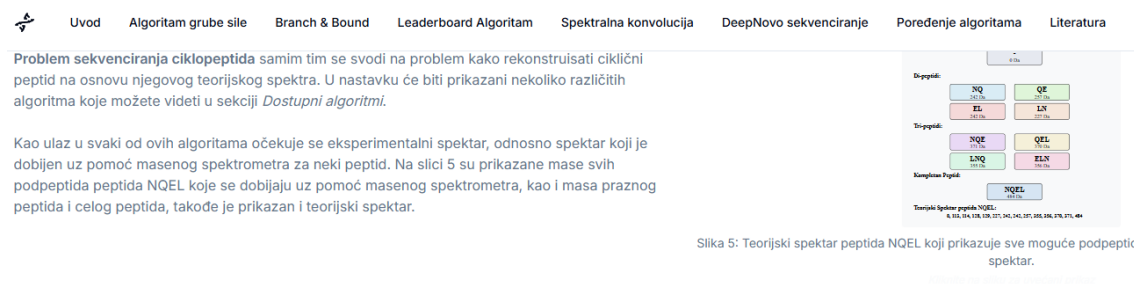


Slika 1: Transkripcija DNK u RNK. Enzim RNK polimeraza se kači na početak gena i kreće kroz gene gde razdvaja lanac i stvara prostor za prepisivanje DNK u RNK čime se dobija RNK.

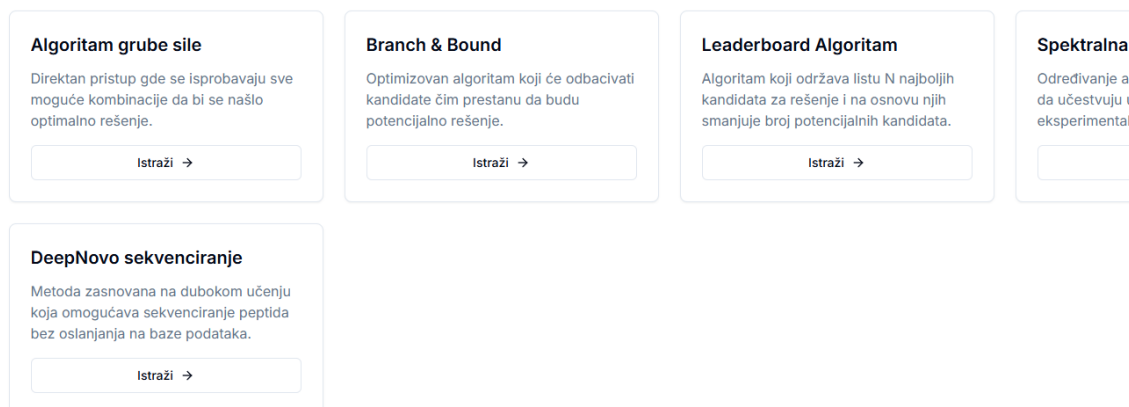


Slika 4.9: Uvodna stranica sa teorijskim pojmovima

GLAVA 4. ELEKTRONSKA PLATFORMA



Dostupni algoritmi



Slika 4.10: Navigacioni meni na dnu Uvodne stranice koji vodi ka algoritmima

4.2.3 Algoritam grube sile

Stranica za algoritam grube sile, na slici 4.11, sadrži objašnjenje ovog algoritma kao i kod u programskom jeziku *Python* za implementaciju algoritma.

Nakon objašnjenja i koda za algoritam može da se unese eksperimentalni spektr i postoji mogućnost da se odabere da li korisnik želi da vidi vizuelizaciju ovog algoritma ili želi da vidi samo rešenja 4.12. Na slici 4.13 može da se vidi drvo koje se izgradilo da bi se došlo do rešenja. Postoje i elementi za kontrolisanje izvršavanja animacije kao što su *Play/Pause/Reset* a pored toga postoji i *slider* kojim može da se premota animacija do određenog dela izvršavanja. Pošto drvo izvršavanja može da bude veoma veliko dodata je i opcija uveličavanja tako da određeni deo drveta može da se uveliča a onda ostatak drveta može da se vidi povlačenjem miša. Čvorovi koji su obeleženi zelenom bojom predstavljaju rešenje a crveni čvorovi su odbačeni i ne predstavljaju rešenje. Kada se miš postavi iznad nekog crvenog čvora prikazaće se i objašnjenje zašto on nije rešenje. Kada se animacija završi moguće je kliknuti i na



Algoritam grube sile

Algoritam grube sile (eng. Brute Force) [2] je najjednostavniji pristup rešavanju problema sekvenciranja antibiotika. Ovaj metod sistematski ispi mogle formirati peptid zadate mase. Iako je ovaj pristup garantovano pronalazi tačno rešenje ako ono postoji, njegova vremenska složenost je čini nepraktičnim za duže peptide.

Na primer, za peptid mase 579 Da, algoritam će generisati sve moguće kombinacije aminokiselina i proveriti da li njihova ukupna masa odgovara postojati različiti peptidi sa istom ukupnom masom (FPAYT i QNWGS), što dodatno komplikuje problem.

F	147 Da	Q
P	97 Da	N
A	71 Da	W
Y	163 Da	G
T	101 Da	S
Ukupna masa:		579 Da
Ukupna masa:		

Da bi se utvrdilo koji od peptida je tačno rešenje, algoritam mora da generiše teorijski spektar za svaki kandidat peptid i uporedi ga sa eksperimentalnim spektrom, ali je neophodno za pronalaženje tačnog rešenja.

Naredni deo prikazuje implementaciju ovog algoritma u programskom jeziku Python.

Kod za algoritam grube sile

Algoritam na ulazu očekuje eksperimentalni spektar uređen rastuće koji uključuje 0 i masu celog peptida koji se sekvencira. Implementacija algoritma grube sile počinje od praznog peptida i u svakom prolasku proširuje peptide dodavanjem aminokiseline uz pomoć

Slika 4.11: Stranica za algoritam grube sile sa objašnjenjem istog

dugme *Download* čime će se izgrađeno drvo skinuti na Vaš računar u *SVG* formatu.

GLAVA 4. ELEKTRONSKA PLATFORMA

ostala prethodjama rešenja, preporuče se da se unese sekvence manje. Na kraju će biti prikazani peptidi koji predstavljaju najbolje kandidate za. Ako želite da ipak unesete neke sekvence koje su duže kliknite na dugme omogućeno brzo prikazivanje samih kandidata za traženi spektar.

Primeri peptida i njihovih teorijskih spektara:

- **G:** [0, 57]
- **GA:** [0, 57, 71, 128]

Unesi sekvencu

☐ Prikaži samo rešenje (bez vizuelizacije)

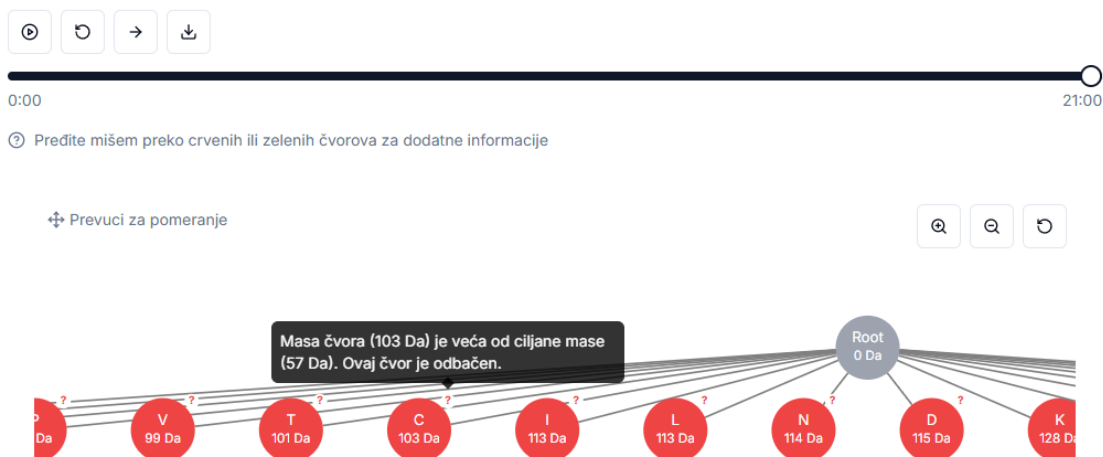
Analiziraj sekvencu

0:00

ⓘ Predite mišem preko crvenih ili zelenih čvorova za dodatne informacije

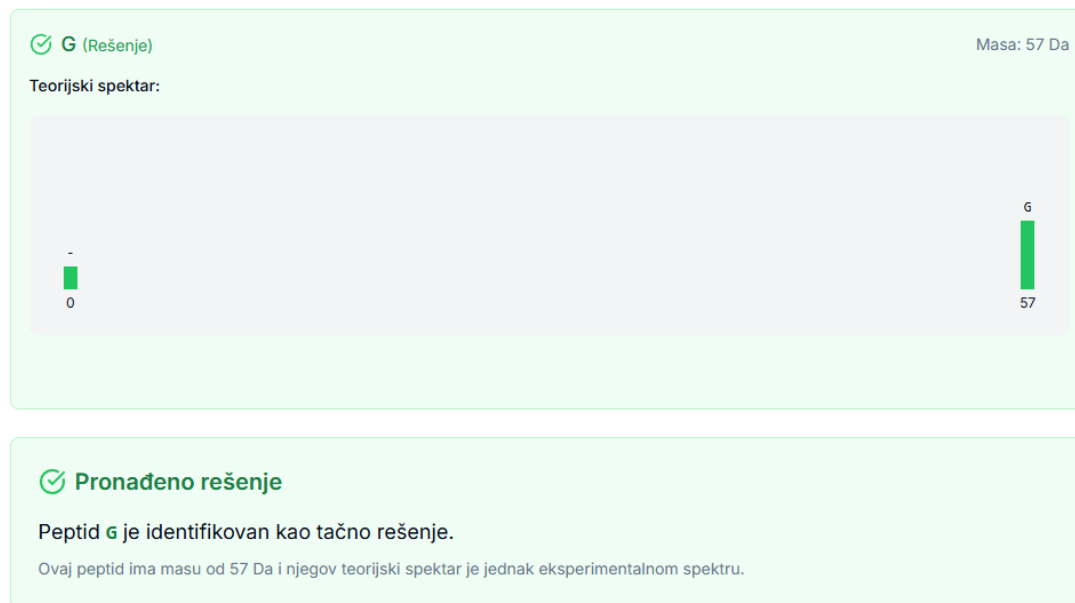
⌵ Prevuci za pomeranje

Slika 4.12: Ulazna forma za eksperimentalni spektar sa opcijom da se označi da li se želi vizuelizacija ili ne



mentalni spektar prikazaće se poruka da je pronađeno rešenje, što se može videti i na slici 4.14. U slučaju da za zadati eksperimentalni spektar nema rešenja ispisaće se poruka koja to i govori.

Kandidati sa masom 57 Da:



Slika 4.14: Pronađena rešenja u algoritmu grube sile

4.2.4 Branch and Bound

Stranica za algoritam *Branch and Bound*, na slici 4.15, sadrži objašnjenje ovog algoritma kao i kod u programskom jeziku *Python* za implementaciju algoritma.

Nakon objašnjenja i koda za algoritam može da se unese eksperimentalni spektar i može da se odabere da se vidi vizuelizacija ovog algoritma ili samo rešenje. Ovaj algoritam je veoma sličan algoritmu grube sile, tako da je i vizuelizacija i elementi koji čine tu vizuelizaciju potpuno ista. Jedina razlika je što je ovaj algoritam dosta brži i može da ranije odseče kandidate koji nisu rešenje tako da eksperimentalni spektar koji može da se unese je duži. Dodatno, pošto u ovom algoritmu postoji više razloga zašto je neki čvor odsečen tako će i poruke koje se prikazuju kada se mišem prevuče preko nekog čvora biti drugačije. Ovde je takođe po okončanju animacije moguće skinuti izgrađeno drvo u *SVG* formatu.



Branch and Bound Algoritam

Branch and Bound algoritam [2] je optimizovana verzija algoritma grube sile koja koristi strategiju "podeli pa vladaj" za efikasnije pretraživanje prostora re ispituje sve moguće kombinacije, Branch and Bound algoritam inteligentno eliminiše delove prostora pretrage koji ne mogu sadržati optimalno rešenje.

U kontekstu sekvenciranja peptida, algoritam funkcioniše na sledeći način:

- **Grananje (Branch):** Algoritam gradi stablo pretrage gde svaki čvor predstavlja delimičnu sekvencu peptida. Svaki čvor se grana dodavanjem nove ami
- **Ograničavanje (Bound):** Za svaki čvor, algoritam procenjuje da li taj put može dovesti do validnog rešenja. Ako masa peptida već premašuje ciljanu ma sekvence peptida nije konzistentan sa eksperimentalnim, ta grana se "odseca" i dalje ne istražuje.
- **Optimizacija:** Algoritam može koristiti dodatne heuristike za procenu koje grane prvo istražiti, što dodatno ubrzava pronalaženje rešenja.

Prednosti Branch and Bound algoritma u odnosu na algoritam grube sile su značajne:

Algoritam grube sile

- Istražuje sve moguće kombinacije
- Eksplozivna vremenska složenost
- Garantuje pronalaženje svih rešenja
- Neefikasan za duže peptide

Branch and Bound

- Inteligentno eliminiše neperspektivne grane
- Značajno bolja vremenska složenost (u najgorem složenosti ali i dalje dosta brži)
- I dalje garantuje pronalaženje svih rešenja
- Efikasniji za duže peptide

Kod za Branch and Bound algoritam


Algoritam na ulazu očekuje eksperimentalni spektar uređen rastuće koji uključuje 0 i masu celog peptida koji se sekvencira. Za razliku od algoritma gr koristi dodatne provere da bi eliminisao neperspektivne grane što ranije.


```
def branch_and_bound(target_spectrum):  
    peptides = []  
    results = []  
    target_peptide_mass = target_spectrum[-1]
```


Slika 4.15: Stranica za algoritam Branch and Bound sa objašnjenjem istog


4.2.5 Leaderboard


Stranica za algoritam *Leaderboard*, na slici 4.16, sadrži objašnjenje ovog algoritma kao i kod u programskom jeziku *Python* za implementaciju algoritma.



[Uvod](#)



[Algoritam grube sile](#)



[Branch & Bound](#)


[Leaderboard Algoritam](#)


[Spektralna konvolucija](#)


[DeepNovo sekvenciranje](#)


[Poređenje algoritama](#)



[Literatura](#)


Leaderboard Algoritam


Leaderboard algoritam [2] je optimizovani pristup za sekvenciranje peptida. Za razliku od sekvenciranja grubom silom koje zahteva tačno poklapanje između teorijskog spektra eksperimentalnog spektra, ovaj algoritam je dizajniran da radi sa **nedostajućim i lažnim masama** tako što prati samo najbolje kandidate peptida umesto svih mogućnosti.

U realnim eksperimentalnim podacima masene spektrometrije, izmereni eksperimentalni spektar je često zašumljen i nepotpun. Neki očekivani fragmenti peptida mogu nedostajati zbog pozadinskog šuma. Savršeno poklapanje između teorijskog spektra peptida i eksperimentalnog spektra je malo verovatno.


Prednosti Algoritma



Efikasnost
 Fokusira se samo na najperspektivnije kandidate, značajno smanjujući vreme izvršavanja



Skalabilnost
 Efikasno radi sa peptidima različitih dužina bez eksponencijalnog rasta vremena


Preciznost
 Održava visoku tačnost uprkos šumu u eksperimentalnim podacima

Primene


Eksperimentalni Podaci
 Idealan za analizu realnih podataka masene spektrometrije sa šumom


Nepotpuni Podaci
 Kada tačno poklapanje nije moguće zbog nepotpunih ili netačnih podataka


Vremenski Kritične Analize
 Kada je potrebna brza identifikacija peptida iz velikih skupova podataka

Kod za Leaderboard algoritam

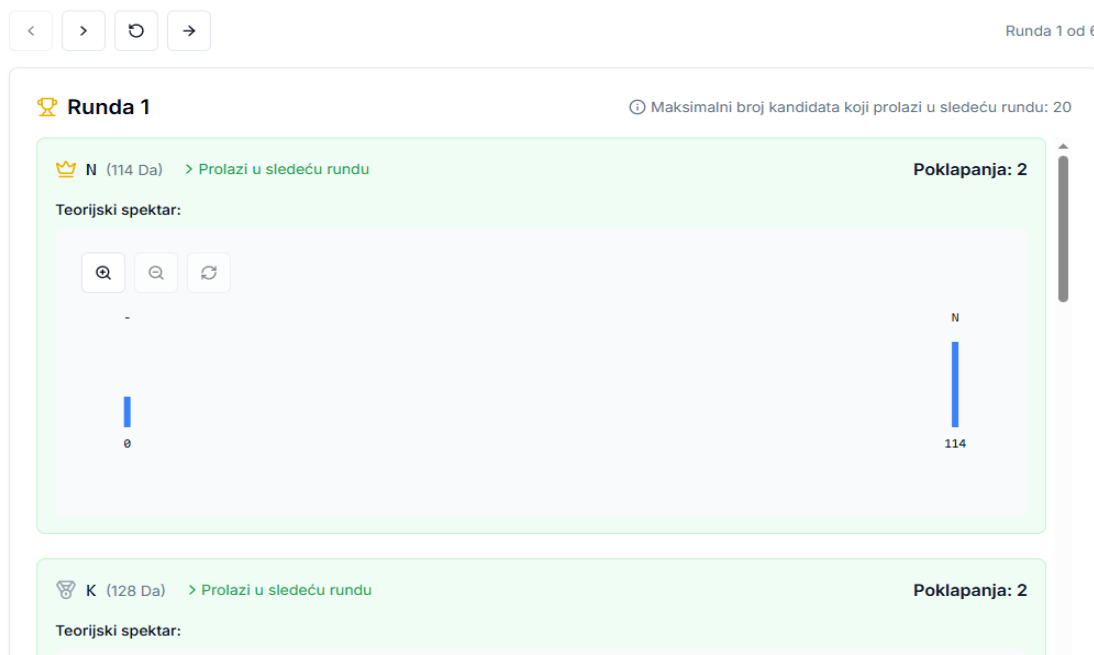
Algoritam održava listu najboljih kandidata (leaderboard) i u svakoj iteraciji proširuje samo najperspektivnije peptide. Ovo značajno smanjuje prostor pretrage u poređenju : sile. Za svaki od peptida koji se generiše određujemo koji je njegov *score*, odnosno broj masa linearnog spektra peptida koji je jednak masama u eksperimentalnom spektru

```
def leaderboard_sequencing(target_spectrum):
    # Krećemo od praznog peptida
    peptides = ['']

    # Peptid koji je trenutno najbolji kandidat i njegov score
    leader_peptide = ''
```

Slika 4.16: Stranica za algoritam Leaderboard sa objašnjenjem istog

Nakon objašnjenja i koda za algoritam može da se unese eksperimentalni spektar i može da se odabere da se vidi vizuelizacija ovog algoritma ili da se prikažu samo rešenja. Pošto ovaj algoritam funkcioniše po principu prolaska u sledeće runde vizuelizacija za ovaj algoritam je tako i odrađena što se može videti na slici 4.17. Za kontrolu animacije postoje strelice za prebacivanje iz runde u rundu, strelica da nas prebaci skroz u poslednju rundu da bismo videli koje je rešenje kao i dugme za resetovanje odnosno prebacivanje na prvu rundu. Uz svaki peptid piše njegova masa, prikazuje se njegov teorijski spektar kao i broj poklapanja teorijskog spektra sa zadatim eksperimentalnim spektrom. Svaki peptid koji prolazi u sledeću rundu je označen zelenom bojom uz tekst da je prošao dalje. Teorijski spektar svakog od peptida može da se uveliča po potrebi i da se koristi *drag and drop* mehanizam da se vidi neki određeni deo spektra. Da bi se poboljšale performanse vizuelizacije za prikaz svih mogućih kandidata u nekoj rundi korišćen je princip *lazy loading*, odnosno tek kada se lista spušta na dole prikazaće se sledeći kandidati koji su razmatrani u toj rundi.



Slika 4.17: Prikaz rundi za Leaderboard algoritam

U poslednjoj rundi biće prikazani peptidi koji su rešenje zadanog eksperimentalnog spektra i to je prikazano na slici 4.18. Za peptide koji nisu rešenje u nekoj rundi prevlačenjem miša preko njih biće prikazana i poruka o razlogu zašto nisu rešenje.

<
>
↺
→

Runda 6 od 6

🏆 Runda 6

✓ **Ukupno različitih rešenja: 52**

NKE Masa: 371 Da Poklapanja: 8	NQE Masa: 371 Da Poklapanja: 8	NEK Masa: 371 Da Poklapanja: 8
NEQ Masa: 371 Da Poklapanja: 8	KNE Masa: 371 Da Poklapanja: 8	KEN Masa: 371 Da Poklapanja: 8
QNE Masa: 371 Da Poklapanja: 8	QEN Masa: 371 Da Poklapanja: 8	ENK Masa: 371 Da Poklapanja: 8
ENQ Masa: 371 Da Poklapanja: 8	EKN Masa: 371 Da Poklapanja: 8	EQN Masa: 371 Da Poklapanja: 8
NEGA Masa: 371 Da Poklapanja: 8	NEAG Masa: 371 Da Poklapanja: 8	KEGG Masa: 371 Da Poklapanja: 8
QEGG Masa: 371 Da Poklapanja: 8	ENG Masa: 371 Da Poklapanja: 8	ENAG Masa: 371 Da Poklapanja: 8

Slika 4.18: Poslednja runda i peptidi koji su rešenje za Leaderboard algoritam

4.2.6 Spektralna konvolucija

Stranica za algoritam spektralne konvolucije, na slici 4.19, sadrži objašnjenje ovog algoritma kao i kod u programskom jeziku *Python* za implementaciju algoritma.



Spektralna konvolucija

Spektralna konvolucija [2] je tehnika koja se koristi za identifikaciju aminokiselina koje mogu biti prisutne u peptidu na osnovu eksperimentalnog spektra. Ova metoda analizira razliku spektra i identifikuje one koje odgovaraju masama aminokiselina.

Proces se sastoji iz dva glavna koraka:

1. **Izračunavanje konvolucije:** Za svaki par masa u spektru, izračunava se njihova razlika. Ove razlike mogu odgovarati masama aminokiselina.
2. **Identifikacija aminokiselina:** Najčešće razlike koje se pojavljuju u spektru verovatno odgovaraju aminokiselinama prisutnim u peptidu. Te aminokiseline se izdvajaju i koriste u *L*-algoritmu.

Glavna prednost ovog algoritma jeste to što u samom startu smanjuje skup aminokiselina koje mogu da učestvuju u građenju peptida, čime se algoritam dosta ubrzava. Takođe, ovi mogućnosti da se identifikuju nepoznate ili modifikovane aminokiseline. Još jedna od prednosti ovog algoritma jeste to što može da radi na eksperimentalnim spektrima koji imaju jo nedostajućih masa.

Kod za algoritam spektralne konvolucije

Algoritam spektralne konvolucije računa razlike između svih parova masa u eksperimentalnom spektru, a zatim identifikuje najčešće razlike koje odgovaraju masama aminokiselina se zatim koriste za generisanje kandidata peptida.

```
def spectral_convolution(target_spectrum):
    num_of_el_in_spectrum = len(target_spectrum)
    convolution = []

    # Elemente spektra možemo da posmatramo kao matricu, gde prvu vrstu i prvu kolonu predstavljaju elementi eksperimentalnog spektra.
    # Prolazimo kroz elemente spektra i međusobno ih poredimo, kao kroz donju trougaonu matricu i ako je razlika u okviru datog opsega onda je to
    # jedna aminokiselina koja može a učestvuje u rešenju
    for i in range(num_of_el_in_spectrum):
        for j in range(i):
            diff = target_spectrum[i] - target_spectrum[j]
            if 57 <= diff <= 280:
                convolution.append(diff)

    # Određujemo broj pojavljivanja masa
```

Slika 4.19: Stranica za algoritam spektralne konvolucije sa objašnjenjem istog

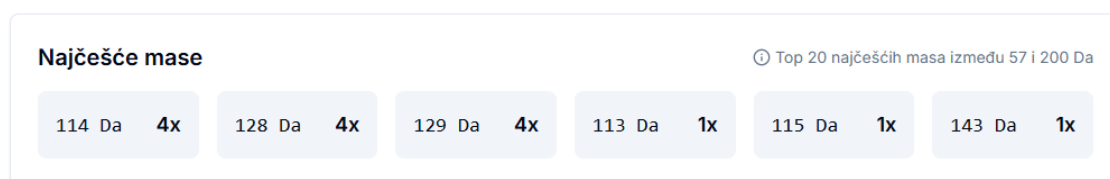
Nakon objašnjenja i koda za algoritam može da se unese eksperimentalni spektar i može da se odabere da se vidi vizuelizacija ovog algoritma ili da se prikažu samo rešenja. Ovaj algoritam je veoma sličan *Leaderboard* algoritmu, tako da je i vizuelizacija i elementi koji čine tu vizuelizaciju potpuno ista. Jedina razlika je što je ovaj algoritam na samom početku kreira matricu konvolucije, slika 4.20, i uz pomoć matrice smanjuje broj aminokiselina koje mogu da učestvuju u izgradnji peptida, slika 4.21. Postoje posebne kontrole za kontrolisanje animacije za izgradnju matrice konvolucije, kao i posebne kontrole za *Leaderboard* deo ovog algoritma. Ove kontrole su slične onima iz prethodnih algoritama.

Matrica konvolucije



Slika 4.20: Matrica konvolucije u algoritmu spektralne konvolucije

Aminokiseline u peptidima



Slika 4.21: Najčešće mase aminokiselina koje se pojavljuju u matrici konvolucije

4.2.7 DeepNovo

Stranica za *DeepNovo* sekvenciranje, na slici 4.22, sadrži *tab*-ove koji pružaju uvid u detalje kao što su:

- **Pozadina** - koji su trenutni problemi sekvenciranja
- **DeepNovo** - šta je DeepNovo i kako funkcioniše
- **Arhitektura** - koje komponente čine DeepNovo
- **Rezultati** - uspešnost primene DeepNovo metode



Metoda za De Novo Sekvenciranje Peptida Zasnovana na Dubokom Učenju

Pristup sekvenciranju peptida koji koristi duboko učenje za identifikaciju sekvenci aminokiselina bez oslanjanja na baze podataka

Pozadina	DeepNovo	Arhitektura	Rezultati
----------	----------	-------------	-----------

Postojeće tehnike

Trenutne tehnike za sekvenciranje peptida (poput pretraživanja baze podataka i de novo sekvenciranja) mogu imati poteškoća u radu sa novim, složenim ili nepotpunim podacima. Pristup pretraživanja baze podataka oslanja se na poređenje eksperimentalnih podataka sa bazom podataka poznatih proteinskih sekvenci, ali to je problem jer:

Nepoznati proteini

Novi proteini koji nikada ranije nisu viđeni neće se podudarati ni sa čim u bazi podataka, što dovodi do nemogućnosti identifikacije.

Nedostajući podaci

Podaci generisani iz eksperimenata masene spektrometrije mogu biti pogrešni i nepotpuni, što otežava pouzdano poređenje.

Pristup de novo sekvenciranja pokušava izgraditi sekvencu peptida iz početka, bez oslanjanja na bazu podataka, ali je često manje precizan i računski skup. DeepNovo kombinuje prednosti oba pristupa koristeći duboko učenje za poboljšanje tačnosti de novo sekvenciranja.

Računska složenost

Slika 4.22: Stranica za DeepNovo sekvenciranje

4.2.8 Poređenje algoritama

Stranica za poređenje algoritama, na slici 4.23, prikazuje vremena izvršavanja algoritama grube sile, *Branch and Bound*, *Leaderboard* i spektralne konvolucije. Dodatno, za svaki algoritam prikazuje i to da je pronašao rešenje i ako da koliko ih je. Na kraju će za svaki od algoritama biti prikazana i njegova rešenja.

Poređenje vremena izvršavanja

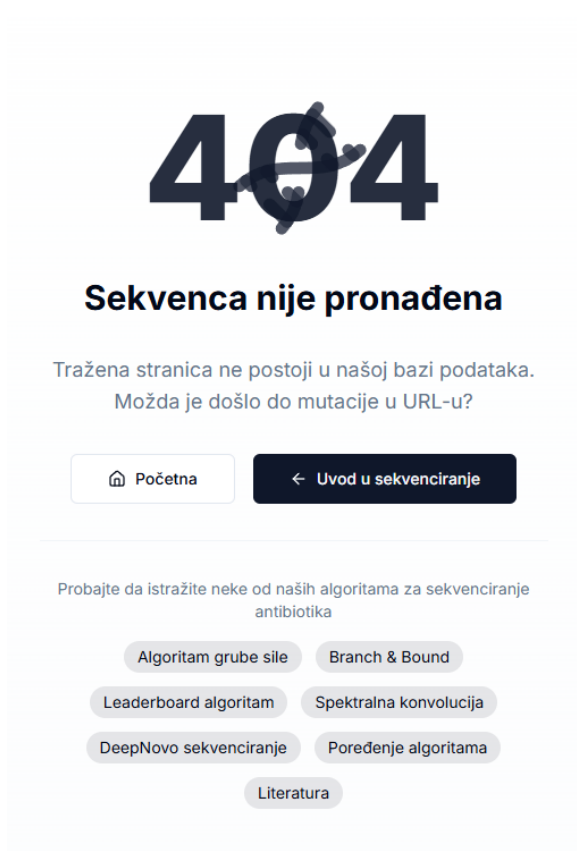
Algoritam	Vreme izvršavanja (u sekundama)	Broj pronađenih rešenja	Status
Algoritam grube sile	6.3618	0	Bez rešenja
Branch and Bound	0.0048	0	Bez rešenja
Leaderboard algoritam	0.0120	24	Uspešno
Spektralna konvolucija	0.0123	24	Uspešno

Analiza performansi
Najbrži algoritam:
Branch and Bound (0.0048)
Ukupno različitih rešenja:
24

Slika 4.23: Poređenje brzine izvršavanja algoritama grube sile, *Branch and Bound*, *Leaderboard* i spektralne konvolucije

4.2.9 Nepostojeća stranica

U slučaju da se unese pogrešan *URL* za stranicu prikazaće se da ta stranica ne postoji i ponudi izbor postojećih stranica, na slici 4.24 može da se to i vidi.



Slika 4.24: Stranica koja se prikazuje u slučaju da je pogrešan *URL* unet

Glava 5

Zaključak

Razvijena elektronska platforma za sekvenciranje antibiotika predstavlja edukativni alat koji objedinjuje teorijske koncepte sa praktičnom implementacijom. Kroz interaktivne simulacije i vizuelizacije, platforma omogućava dubinsko razumevanje kompleksnih algoritama sekvenciranja.

Glavni doprinosi ovog rada uključuju:

- Integraciju više algoritama za sekvenciranje u jedinstvenu platformu
- Vizuelizaciju koraka algoritama u realnom vremenu
- Kreiranje interaktivnog okruženja za učenje
- Pобољшanje dostupnosti bioinformatičkih alata studentima

Budući radovi mogu se usredsrediti na proširenje platforme sa dodatnim algoritmima, poboljšanje performansi postojećih implementacija i integraciju sa stvarnim eksperimentalnim podacima iz masene spektrometrije.

Bibliografija

- [1] Django dokumentacija, 2025. on-line at: <https://docs.djangoproject.com/en/5.2/>.
- [2] Docker Compose dokumentacija, 2025. on-line at: <https://docs.docker.com/compose/>.
- [3] Docker dokumentacija, 2025. on-line at: <https://docs.docker.com/manuals/>.
- [4] GitHub repozitorijum sa izvornim kodom aplikacije, 2025. on-line at: <https://github.com/Milak9/master>.
- [5] Google Cloud Run dokumentacija, 2025. on-line at: <https://cloud.google.com/run/docs/overview/what-is-cloud-run>.
- [6] NextJS dokumentacija, 2025. on-line at: <https://nextjs.org/docs>.
- [7] Node 18 dokumentacija, 2025. on-line at: https://devdocs.io/node~18_lts/.
- [8] Python 3.12 dokumentacija, 2025. on-line at: <https://docs.python.org/3.12/index.html>.
- [9] TypeScript dokumentacija, 2025. on-line at: <https://www.typescriptlang.org/docs/>.
- [10] Pavel Pevzner Ari Frank. PepNovo: De Novo Peptide Sequencing via Probabilistic Network Modeling. *Analytical Chemistry*, 77:964–973, 2005.
- [11] Muhammad Zubair Eshita Garg. Mass Spectrometer, 2024. on-line at: <https://www.ncbi.nlm.nih.gov/books/NBK589702/>.

- [12] Jovana Kovačević. Materijal sa predavanja „Uvod u bioinformatiku”, 2022. on-line at: https://www.bioinformatika.matf.bg.ac.rs/predavanja/Chapter_4.pdf.
- [13] Bin Ma. Novor: Real-Time Peptide de Novo Sequencing Software. *Journal of The American Society for Mass Spectrometry*, 26:1885–1894, 2015.
- [14] Pavel Pevzner Philip Compeau. Bioinformatics Algorithms: An Active Learning Approach Vol. I, Chapter 4: How Do We Sequence Antibiotics?, 2015. on-line at: <https://cogniterra.org/course/64/promo>.
- [15] Sanju Tamang. Codon Chart: Table, Amino Acids & RNA Wheel Explained, 2024. on-line at: <https://microbenotes.com/codon-chart-table-amino-acids/>.
- [16] Jing et al. Zhang. PEAKS DB: De Novo Sequencing Assisted Database Search for Sensitive and Accurate Peptide Identification. *Molecular & Cellular Proteomics*, 11, 2011.
- [17] Xin et al. Zhang. DeepNovo: De novo peptide sequencing by deep learning. *PNAS*, 114:8247–8252, 2017.

Biografija autora

Miloš Milaković rođen je 6. avgusta 1998. godine u Beogradu. Smer Informatika na Matematičkom fakultetu Univerziteta u Beogradu upisao je 2017. godine, a završio 2021. godine sa prosečnom ocenom 8.54. Nakon toga je upisao master studije na istom smeru. Od septembra 2021. do marta 2024. godine je zaposlen na poziciji *Software* developer u firmi **Endava**. Od marta 2024. godine do sada je zaposlen na poziciji *Software* developer u firmi **LotusFlare**. Projekti na kojima je radio su uglavnom bili zasnovani na veb tehnologijama (*Telco* industrija i *FinTech* industrija), a osnovni programski jezik u kojem su projekti rađeni su *Python*, *Lua* i *TypeScript*.